Tracking control for quadrotors in global position and heading direction

Master's Thesis
DC 2022.044

Author:
T.J.M. Snijders

Supervisor:
dr.ir. A.A.J. Lefeber

This report was made in accordance with the TU/e Code of Scientific Conduct for the Master thesis.

Department of Mechanical Engineering
Master Mechanical Engineering
Dynamics And Control Section

Friday 20th May, 2022

Summary

As drones, or unmanned aerial vehicles (UAVs), become a more predominant solution to problems in today's society, so are the questions as to what they can achieve in the near future. In this near future, it could be possible that drones are deployed for monitoring services in either or both the surveillance or agriculture industry. Tracking control algorithms are crucial for this to happen. This thesis attempts to contribute to solving tracking control problems for a specific drone: the quadrotor. Earlier work on this topic already resulted in good performance, although there were still improvements to be made. This thesis' main objective therefore, is to improve on the earlier work on this topic, because it still showed undesirable transient behaviour in position and heading direction. It specifically continues on the work of Lefeber, van den Eijnden, and Nijmeijer (2017) and Lefeber, Greiff, and Robertsson (2020). Hereto, their position control laws are investigated, to validate if they lead to the undesirable transient behaviour. The attitude control and combined control are investigated for the same reason as well. The results from these validations show the cause of the undesirable behaviour, after which improvements for these problems are proposed. To improve the heading direction tracking performance, a new reference definition is proposed. This new reference definition is dependent on the actual global position and a target reference position, instead of time, which is useful for monitoring and surveillance purposes. Numerical experiments show that the newly designed control laws perform better during transient behaviour when compared to the earlier work. These experiments also show some flaws in the newly designed reference heading direction method, for which recommendations for improvements are proposed.

Acknowledgements

I would like to express my gratitude to multiple people for their support during the period I have worked on this thesis and during the whole of my study.

First of all, I would like to thank my mentor, coach and supervisor dr. ir. A.A.J. Lefeber. Erjen, thank you for your support, patience, confidence and insight during the period of my thesis, but also before that. I will not forget that it was you who designed the perfect assignment for me and, because of that, I can say I have had an enjoyable time.

I also want to thank my family for their (financial) support and endless amounts of love, without which I probably would have never made it this far.

Next, I would like to thank my "makkers" from on and off the university for their necessary and precious distractions during my studies.

And finally, a special word to my girlfriend. Dear Iris, Thank you for your encouragement, for always putting a smile on my face and for your patience. Hopefully, this can be the beginning of our time.

Tommy Snijders
May 17, 2022

Contents

Nomenclature

Reference frames

$\mathcal{B}$	Right-handed ortonormal body frame in NED configuration
$\mathcal{D}$	Right-handed ortonormal desired direction frame in NED configuration
$\mathcal{I}$	Right-handed ortonormal inertial frame in NED configuration
$\mathcal{R}$	Right-handed ortonormal reference frame in NED configuration

Number sets

$\mathbb{N}$	The set of natural numbers
$\mathbb{R}$	The set of real numbers
$\mathbb{Z}$	The set of integer numbers
$\mathfrak{so}(n)$	The set of $n \times n$ skew symmetric matrices
$SO(n)$	The n-dimensional special orthogonal group

Operators

$\det(\cdot)$	Determinant
$\dot{x}, \ddot{x}$	First and second derivative of x
$\frac{\partial}{\partial x}$	Partial derivative with respect to x
$\hat{x}$	Estimate of x
$\sigma(\cdot)$	Saturation function
$\tilde{x}$	Error between actual value and estimate value of x
$d(\cdot, \cdot)$	Logarithmic map log between two rotation matrices
L_f	Lie derivative of function f
$S(\cdot)$	Skew symmetric matrix
$x^\top, A^\top$	Transpose of vector x or matrix A

Symbols

δ	Innovation terms
ϵ	Scalar bound
η	Internal part state space in nominal form
γ	Scalar value for saturation function
λ	Angle of the actual thrust vector, the desired thrust vector or the difference between them in a frame rotated around f_d
ν	Linear body-fixed velocity, with respect to $\mathcal{B}$ and $\mathcal{R}$
Ω	Angular velocity of propeller
ω	Angular velocity, with respect to $\mathcal{B}$ and $\mathcal{R}$
ϕ	Roll angle
ψ	Yaw angle
ρ	Position of origin of $\mathcal{B}$ and $\mathcal{R}$, with respect to $\mathcal{I}$
τ	Torque
θ	Pitch angle
ξ	External part state space in nominal form
C, c	Matrix control terms, scalar control terms
d	Scalar damping parameter, or disturbance
f	Thrust
g	Gravitational acceleration constant

I	Identity matrix
J	Inertia matrix
k	Relative degree, scalar value for mu or control value
l	Distance to center of mass of quadrotor
m	Mass
n	State dimension
T	Thrust value of propeller
t	Time
u	(virtual) Input
v	Unit vectors or state feedback control
x	State of system
y	Output
z	Angle of the desired thrust vector in the horizontal plane from the front of the quadrotor

Subscripts

0	Initial condition
$\mathcal{B}, \mathcal{D}, \mathcal{I}, \mathcal{R}$	Reference frames
ω	State ω
d	Desired value
e	Error value
f	Most recently feasible value
i	Numerator
R	State R
r	Reference
RP	Roll and pitch
t	Target reference position for ρ
u	Rotated frame in which virtual input u can be described in two directions
v	Constant scalar value

Acronyms

GPS	Global positioning system
IMU	Internal measurement unit
NED	North-East-Down, configuration for attitude frames
PWM	Pulse width modulation
RPY	Roll-Pitch-Yaw, ZYX Euler angle sequence
U(a)GAS	Uniform (almost) global asymptotic stability
UAV	Unmanned aerial vehicle
UGES	Uniform global exponential stability
ULES	Uniform local exponential stability

Chapter 1

Introduction

1.1 Background

Unmanned Arial Vehicles (UAVs), also referred to as drones, have been around for more than a century [3]. Where its applicability in the earlier years was focused on mainly warfare, it has now shifted towards research and civilian applications over the past few decades. Commercially available drones made for video and photography now allow users to capture images that otherwise would not be possible, due to the great heights and distances a drone can go without losing connectivity. Drones are now also able to assist researchers by flying in hazardous areas to collect data that would otherwise be unobtainable. For example, a drone was able to fly autonomously over an erupting volcano in Bali, Indonesia to collect data which in turn could be used to predict the trajectories of lava streams [4]. Knowing the past and expected growth of the drone market worldwide, it is expected that developments like this steadily continue in the coming years and it can only be imagined what that would mean for the near future [5].

When thinking about futuristic smart cities, UAVs also play a key role. Figure 1.1, shows some examples of what UAVs might be able to do in the foreseeable future. Where some of these examples, say a UAV taxi, could still be unfeasible for multiple years, others might be more realistic than one might think. Amazon, for instance, is one of the companies that is investing heavily in parcel delivery by UAVs, as a new Amazon Air hub is recently launched in the U.S. [6]. Zipline on the other hand, has already implemented autonomous drones in Ghana, where they were able to shuttle medical supplies to suspected COVID-19 patients [7]. Another example can be found in agriculture, where drones can already be used in the monitoring of crops or livestock and in the spraying of fruits, vegetables, and trees [8]. A closer look into the research papers of the earlier mentioned examples

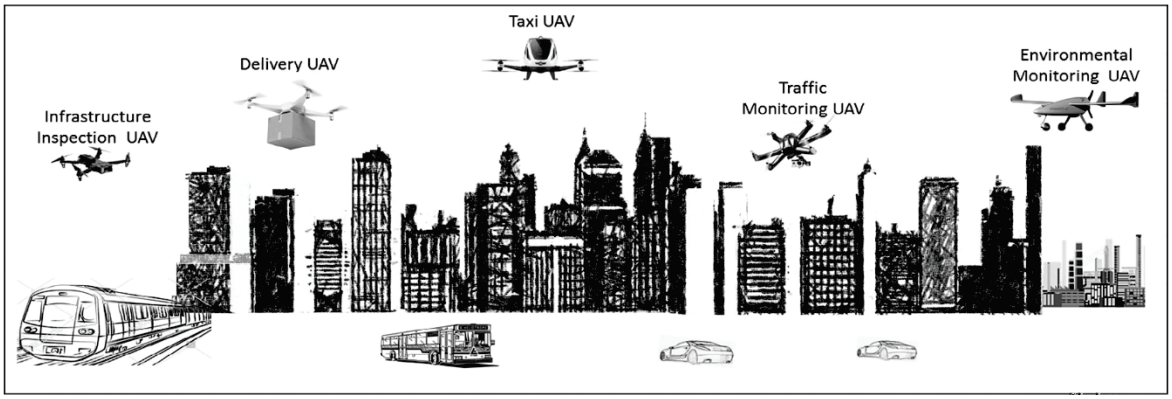


Figure 1.1: *Examples of applications for UAVs in smart cities [9]*

reveals that different situations require different UAVs. In case of the UAV in Bali, a fixed-wing model, see Figure 1.2a, was chosen because of its stronger aerodynamics, longer flight duration, bet-

ter flight control system, higher flight safety, and larger coverage area [10], [11]. However, most of the drones used for monitoring and spraying crops are multi-rotor winged. There are different types of multi-rotor winged UAVs, where the differences lie in the number of rotors. Some examples are quadrotors, see Figure 1.2b, or octocopters. Reasons to use a multi-rotor winged UAV can be high maneuverability, the capability of hovering and the ability to vertically take-off and land. Despite that, since multi-rotor winged UAVs are also relatively lightweight, compact and low-priced it makes them the most popular drones for hobby and recreational use [12].

(a) *Fixed-wing drone* **(b)** *Quadrotor from DJI*

Figure 1.2: *Two examples of different drone types*

The next big step in drone research now lies in autonomous flying, without a global positioning system (GPS). Autonomous flight can be achieved by formulating and solving a trajectory tracking problem where the drone translates along a predefined path, without external meddling. Zipline was already able to achieve this, however they used a fixed-wing drone. This research aims to contribute to the goal of autonomous flying without GPS by using a specific quadrotor, the Parrot Mambo Fly, and by tracking four degrees of freedom: the three translational degrees of freedom and one rotational degree of freedom. Additionally, this quadrotor should be able to stay headed towards a specific target point continuously, to also contribute to achieve UAV traffic monitoring or UAV environmental monitoring for the future smart city depicted in Figure 1.1.

1.2 Previous research

In previous research, quadrotor dynamics have been parametrized using different methods. Some of these methods are: rotation matrices, Euler angles and unit quaternions. Reasons to use either method is often a trade-off between fast computing power and little ambiguity. Using quaternions is a method to ensure fast computation speed when multiplying rotations [13]. A downside can be the ambiguity that comes with using quaternions, since a rotation, q, can be described in two different ways, q and $-q$. This can lead to undesirable results when the control system calculates different control efforts for these same rotations. Rotational matrices can be described by orthogonal matrices of size $n \times n$ where $n \in \{2, 3\}$, for respectively planar or spatial rotations. To calculate sequential rotations, the rotational matrices can be multiplied with respect to the order in which these sequential rotations occur. These calculations however, are relatively more demanding than those needed for quaternion multiplications [14].

Lefeber, van den Eijnden, and Nijmeijer (2017) proposed a full state feedback control framework that has shown to almost-globally uniformly asymptotically stabilize the complete closed-loop system and where the estimated errors uniformly locally exponentially converge to the actual errors. Jeurgens (2017) was able to demonstrate the effectiveness of this control structure on a Parrot AR Drone 2.0 in a hovering experiment. The absence of damping effects limited the accuracy of the mathematical model. Brekelmans (2019) was able to include these damping forces into a simulation model for a Parrot Mambo drone, but did not use them to improve on the control framework. Lefeber, Greiff, and Robertsson (2020) extended the research from Lefeber, van den Eijnden, and Nijmeijer (2017) by proposing a novel output feedback controller, to counter the absence of linear velocity mea-

surements. Some of this controller's advantages were that it was able to filter acquired measurements, which could therefore attenuate the effect of measurement noise. Also, this controller was proven to be uniformly locally exponentially stable, making it robust to disturbances. In addition, they were able to parametrize the reference dynamics in $SO(3)$, using the theory of flat outputs introduced by [16], the reference dynamics included a yaw trajectory as well. The concept of flat output is discussed in more detail later in this report.

1.3 Research gap

However, there remains a gap in research into the ability to use this knowledge to continuously and accurately track a yaw reference which aims towards a specific target, while following a reference trajectory, in transient behaviour. Transient behaviour describes the behaviour of the drone before it has converged to the references. Improving the transient behaviour while tracking a reference trajectory is therefore also the main interest of this thesis. Applying a flat output trajectory as in [2], and the control laws and parameters from [1], results in correct, but undesired tracking during transient behaviour. The following examples demonstrate this. In these examples, the reference trajectory is defined as

$$\rho_r = \begin{bmatrix} \cos(\omega_v) & \sin(\omega_v) & 1.5 \end{bmatrix}^\top \tag{1.1a}$$

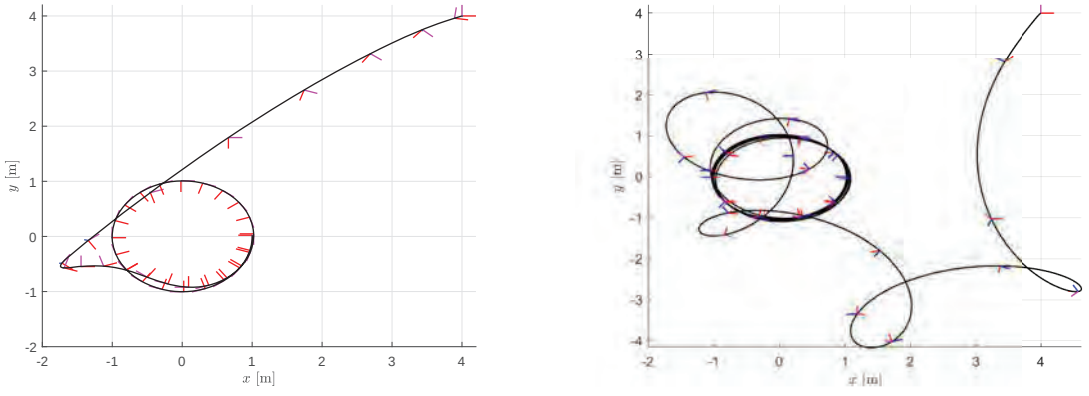
$$\psi_r = -\omega_v t + \pi, \tag{1.1b}$$

with initial conditions

$$\rho(t_0) = \begin{bmatrix} 4 & 4 & 1.5 \end{bmatrix}^\top \qquad \nu(t_0) = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^\top \qquad R(t_0) = I \qquad \omega(t_0) = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^\top. \tag{1.2}$$

The parameters ρ, ψ ν, R and ω are discussed in more detail in Chapter 2.

As visualized in Figure 1.3a, one can see a rotational movement around its z-axis with the same angular velocity as the rotation of the global position, but instead of facing the centre of the circle during transient behaviour, it rotates as if on the correct position already. Even though this behaviour is undesirable in this thesis, since it does not face a certain position constantly, it is the expected behaviour. The reason for this comes from the yaw reference being time dependent, instead of dependent on the position. What can also be observed are the translations during transient behaviour, these translations are fast and aggressive, see for example the quick maneuver in Figure 1.3a around $[x, y] = [-1.75, -0.5]$. This maneuver is a result of the created overshoot from the reference trajectory, as seen in Figure 1.4, from rotating, and therefore translating, too quickly. As impressive that these maneuvers are, they are also the cause of the undesirable transient behaviour of the drone. Similar undesired transient behaviour is seen when increasing the velocity in ρ_r from (1.1a).



(a) $\omega_v = 0.2\pi$. **(b)** $\omega_v = 1.2\pi$.

Figure 1.3: *Quadrotor position and orientation in two-dimensional space while tracking reference dynamics.*

Figure 1.3b clearly shows that the drone tries to follow the trajectory already, without being at the correct position. Again, not only is the orientation incorrect, but also the position.

The behaviours showed in both Figure 1.3 can cause complications, for example when its use lies in surveillance objectives.

Another unwanted consequence which results from the controller of Lefeber, van den Eijnden, and Nijmeijer (2017), is that large initial errors create errors in the z direction while going through transient behaviour, despite the absence of initial errors in this z direction, see Figure 1.4. This too can be an issue for surveillance, as the height and the heading direction both determine the output of a possible frontal camera. The background information given above highlights some of the issues for

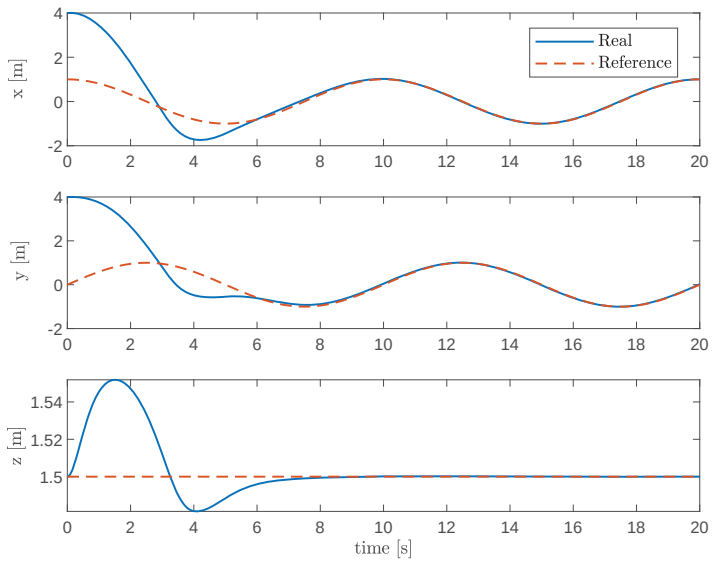


Figure 1.4: *Quadrotor position while tracking the reference trajectories from* (1.1) *with* $\omega_v = 0.2\pi$.

which UAVs can be the solution, as of today and possibly in the future. Also, the main problem for this research is mentioned, and visualized using figures 1.3a and 1.4. The following section provides more detail to the objectives which are to be achieved in this thesis.

1.4 Objectives

For this research, the goal is thus to design a control law that enables a quadrotor to fly autonomously in not only horizontal and vertical direction, but also in heading direction without losing sight off a specific target during transient behaviour. Additionally, this control law should be continuous and should converge to the reference position in a near straight path. This last step is an addition to the research conducted by Lefeber, van den Eijnden, and Nijmeijer (2017) and Lefeber, Greiff, and Robertsson (2020). As shown in Section 1.3, the tracking performance during transient behaviour is not as desired. If the drone could continuously face a certain target, even through transient behaviour, surveillance or monitoring applications would be feasible since the output of the frontal camera can now be utilized continuously. This research is also built on the work of Brekelmans (2019), since a Parrot Mambo Fly was used in his research, which is the same drone as the one used in this thesis.

To conclude that the research has been successful, several sub-objectives are considered, besides the main objective mentioned at the start of this section. These are formulated as follows:

- Investigate the cause of the undesirable transient behaviour, by validating the control laws designed in Lefeber, van den Eijnden, and Nijmeijer (2017) and Lefeber, Greiff, and Robertsson

(2020). Hereto,

  - first validate the position control laws, subsequently
  - validate the attitude control laws, and finally
  - validate the combined control laws.

- Extend and/or improve on the already designed continuous full-state feedback tracking controller from Lefeber, van den Eijnden, and Nijmeijer (2017) and Lefeber, Greiff, and Robertsson to fit desired position and attitude tracking through transient behaviour.
- Embed the controllers in a simulation environment and validate its performance.
- Use the knowledge provided by Brekelmans (2019) to implement the designed controllers into the drone's hardware and check its actual effectiveness.

Note that results from simulations or experiments can prove the ineffectiveness of a designed control law, therefore it might be necessary to take a step back and re-evaluate the designed control law.

1.5 Thesis outline

This thesis has the following format. In Chapter 2 some preliminary notations, control designs, control methods and definitions are provided. The information discussed in this chapter is used extensively throughout this thesis. Chapter 3 is focused on validating the position control laws from earlier research. Hereto, multiple new controllers are designed which should behave as desired, after which these new controllers are compared to the earlier ones. Next, in Chapter 4 a similar approach is followed as in Chapter 3, but now to validate the attitude controllers from earlier literature. Besides that however, a new attitude control law is proposed which is based on a feedback linearization method. Using this new method, possibilities arise to include yaw tracking. Hence, a specific yaw tracking reference is proposed. Then, in Chapter 5, the final controllers from Chapter 3 and 4 are combined to validate if the thrust definition can be improved to increase the tracking performance during transient altitude translations. Different proposals are made and examined for their respective performance. Chapter 6 presents the effectiveness of the designed controllers. These controllers are compared to results obtained by earlier literature so as to find their specific strengths and shortcomings. Lastly, Chapter 7 presents the conclusions of this work and the recommendations for possible future work. Appendix A provides additional information regarding the new method described in Chapter 4 and Appendix B describes the method to generate references from a set of flat outputs.

Chapter 2

Preliminaries

During the whole of this thesis, definitions, relations and results from previous research are used. These results range from commonly known to nontrivial. This chapter provides insight to the non-trivial definitions, relations and results. First of all, the two most commonly used methods in this thesis for parametrizing orientations are discussed. Next, the cascade control structure from [1] is discussed together with the mathematical model of the quadrotor dynamics as they are the basis of this thesis. Subsequently, the attitude control law from Lefeber, Greiff, and Robertsson (2020) is shown. This is followed by a specific form of feedback linearization, which is discussed thoroughly further on in this thesis. After that, some final definitions are presented.

2.1 Attitude representations

In order to represent relative orientations, or attitudes, of a quadrotor with respect to another rigid body, coordinate frames are associated to each body. The geometric relationships between each coordinate frame can then be parametrized by several methods. Each of these methods comes with its distinct strengths and weaknesses. The three most commonly used parametrizations are rotation matrices, Euler angles and quaternions. This section provides information regarding the first two parametrization methods, as they are used throughout this thesis. For more insight regarding quaternions, one is referred to [17].

2.1.1 Rotation matrix

Rotation matrices define the coordinate vectors for the axis of one coordinate frame with respect to another coordinate frame. These matrices have a size of $n \times n$ where $n \in \{2, 3\}$, for respectively planar or spatial rotations and for this thesis only right-handed coordinate frames are assumed. This means that the positive direction of rotation is defined by a clockwise rotation and vice versa. Rotation matrices preserve the lengths and positive directions of rotations while mapping the vectors for the axis between coordinate frames. These rotation matrices are denoted by R with as subscript the direction between coordinates frames. For instance, the rotation matrix that maps vectors in frame A to frame B is denoted by R_{AB}. Rotation matrices are part of the special orthogonal group, $SO(n)$ where n again defines the order of the system, which comes with the following properties:

- $R^{-1} = R^T$
- The columns (and therefore the rows) of R are mutually orthogonal.
- Each column (and again therefore each row) of R is a unit vector.
- $\det(R) = 1$

The main advantage of rotation matrices is that their attitude representations are unique. Another advantage is the capability to sequentially multiply multiple rotation matrices to end up at the desired coordinate frame. A rotation matrix R_{AD} can for example be described by sequentially multiplying

$R_{AB}R_{BC}R_{CD}$. A disadvantage is that these matrix calculations require more computing power than, for instance, quaternion multiplications.

2.1.2 Euler angles

Named after the famous Swiss mathematician, Euler angles are the most commonly used method to represent the attitude in a certain frame [18]. By rotating along three consecutive angles, about the successively rotated axes of the base frame, a relation can be constructed that maps the base frame to the new frame. The order of rotation matters, since generally, sequential rotations along varying axes do not result in identical mappings [19]. Multiple conventions are known when using Euler angles, most common are proper Euler angles, Tait-Bryant angles and Roll-Pitch-Yaw (RPY) angles. The latter is most commonly used for automotive, air and spacecraft, and naval purposes [20]–[22]. For RPY-angles, the order of body-fixed rotations is first along the x axis (ϕ), then along the y axis (θ) and finally along the z axis (ψ). The RPY-angle convention is used extensively throughout this thesis and is presented in the following equation

$$R = R_{\mathcal{BI}} = \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \sin(\phi) & \cos(\phi) \end{bmatrix}. \tag{2.1}$$

This rotation matrix maps the inertial frame, $\mathcal{I}$, to the body frame, $\mathcal{B}$. To obtain the map from the body frame to the inertial frame, the transpose of (2.1) can be used, such that

$$R_{\mathcal{IB}} = R_{\mathcal{BI}}^{\top} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & \sin(\phi) \\ 0 & -\sin(\phi) & \cos(\phi) \end{bmatrix} \begin{bmatrix} \cos(\theta) & 0 & -\sin(\theta) \\ 0 & 1 & 0 \\ \sin(\theta) & 0 & \cos(\theta) \end{bmatrix} \begin{bmatrix} \cos(\psi) & \sin(\psi) & 0 \\ -\sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{2.2}$$

The frames $\mathcal{I}$ and $\mathcal{B}$, together with their respective axes, are visualized in Figure 2.1. To uniquely define the rotation matrix, this thesis assumes $\phi \in \left(-\frac{\pi}{2}, \frac{\pi}{2}\right]$, or $\cos\phi \geq 0$. Without this assumption, similar rotation matrices are found when using a yaw of $\pi + \psi$, a pitch of $\pi - \theta$ and a roll of $\pi + \phi$. Appendix A.3 presents a pseudo code to ensure this assumption holds. These Euler angles come with little ambiguities, as they are intuitive and requires minimal parameters. However, they also come with singularities such as Gimbal Lock [23]. Gimbal Lock arises when $\theta = (n + 1/2)\pi$ and yields

$$R(\psi, ((n + 1/2)\pi, \phi) = \begin{bmatrix} 0 & \sin(\phi + \psi) & \cos(\phi - \psi) \\ 0 & \cos(\phi - \psi) & -\sin(\phi + \psi) \\ 1 & 0 & 0 \end{bmatrix}, \tag{2.3}$$

using the double angle formulae. From this matrix it can be seen that the roll and pitch angles, and therefore also their derivatives, are not uniquely defined anymore, thus causing singularities.

2.2 Cascade control

This thesis builds upon the cascade control structure for quadrotors from [1]. So, in order to understand what is to come, first the theory from earlier research is revisited. First of all, the dynamics for a quadrotor are discussed, subsequently the control laws from [1] and finally the attitude control law from [2].

2.2.1 Quadrotor dynamics

Figure 2.1 already provided the inertial frame and the body frame and the necessary mapping between the two by R. From this figure also ρ can be noticed, which defines the position of the center of mass of the quadrotor with respect to the inertial frame and is expressed by $\rho = (x, y, z)^{\top} \in \mathbb{R}^3$. The body-fixed linear velocities and the body-fixed angular velocities are described by $\nu \in \mathbb{R}^3$ and $\omega \in \mathbb{R}^3$ respectively.

The expression for the quadrotor velocity can be obtained through some logical reasoning, the body-fixed accelerations are derived by using Newton's second law, the attitude kinematics through use of

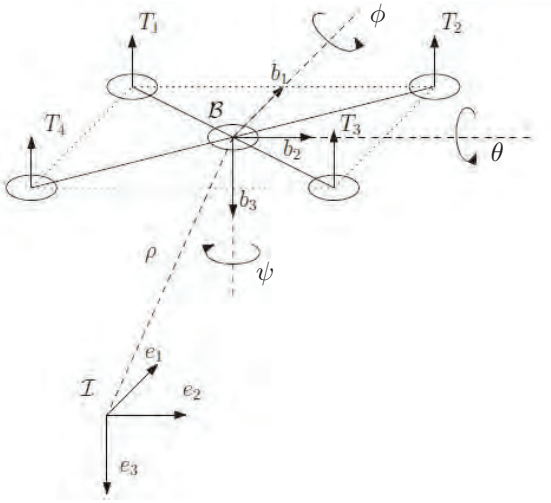


Figure 2.1: *Schematic overview of the quadrotor.*

the Poisson equation and the attitude dynamics again with Newton's second law. For more elaborate explanations on the quadrotor dynamics, one is referred to [12]. The complete quadrotor dynamics can be summarized as follows:

$$\dot{\rho}_{\mathcal{I}} = R_{\mathcal{BI}}\nu_{\mathcal{B}} \tag{2.4a}$$
$$\dot{\nu}_{\mathcal{B}} = -S(\omega_{\mathcal{B}})\nu_{\mathcal{B}} + gR_{\mathcal{BI}}^{\top}e_{3\mathcal{I}} - (f/m)e_{3\mathcal{B}} \tag{2.4b}$$
$$\dot{R}_{\mathcal{BI}} = R_{\mathcal{BI}}S(\omega_{\mathcal{B}}) \tag{2.4c}$$
$$J\dot{\omega}_{\mathcal{B}} = S(J\omega_{\mathcal{B}}) + \tau_{\mathcal{B}}, \tag{2.4d}$$

where S is a skew-symmetric matrix defined as

$$S(a) = -S(a)^{\top} = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix}, \tag{2.5}$$

which is used to write cross products into a compact, clear form. The following notation for a cross product operator is therefore adopted

$$a \times b := S(a)b. \tag{2.6}$$

The quadrotor reference dynamics are described in an identical way. Note that the reference dynamics can be distinguished from the quadrotor dynamics through the subscript $_r$. From these dynamics, a cascaded structure can be recognized, because the position dynamics are influenced by the attitude dynamics through R and ω. This cascaded structure is illustrated in Figure 2.2. To convert the total

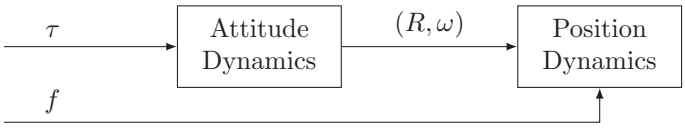


Figure 2.2: *Cascaded structure of quadrotor dynamics.*

force and torques to the force per rotor using the frames from Figure 2.1 the following relation can be used from [13]

$$\begin{bmatrix} f \\ \tau_1 \\ \tau_2 \\ \tau_3 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ l & -l & -l & l \\ l & l & -l & -l \\ -d & d & -d & d \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \end{bmatrix}, \tag{2.7}$$

where l is the distance from the center of a rotor to the center of mass of the quadrotor, d a scalar damping parameter and T_i the thrust per rotor i.

2.2.2 Tracking control

In order to achieve closed-loop stability, [12] proposed a cascaded control structure from which he designed his control laws. The key structure is presented in Figure 2.3. The exact control laws from his thesis are not discussed, as other improved control laws have been developed since then. Lefeber, van den Eijnden, and Nijmeijer (2017) introduced a method to express the translational

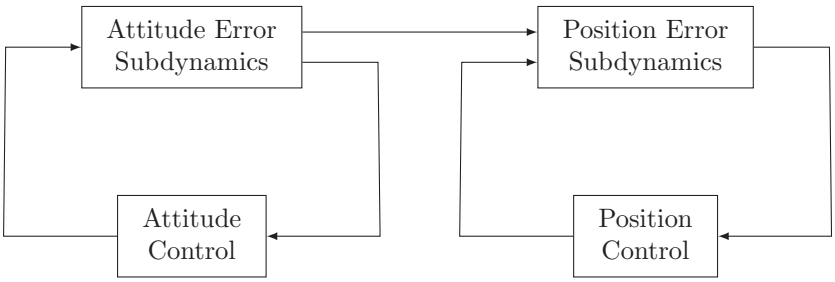


Figure 2.3: *Closed-loop cascade control structure from [1].*

error coordinates in the body frame $\mathcal{B}$, such that the body-fixed linear accelerations can be used as virtual input. An advantage of this method is that independence of the choice of inertial frame is obtained, which would not be the case when defining the translational error coordinates in the body frame. Before obtaining the position tracking error dynamics, first the position tracking error coordinates are defined

$$\begin{bmatrix}\rho_e \\ \nu_e\end{bmatrix} = \begin{bmatrix}R_r^\top(\rho_r - \rho) \\ \nu_r - R_r^\top R\nu\end{bmatrix}, \tag{2.8}$$

where for sake of clarity, the notations $\mathcal{I}$ and $\mathcal{B}$ are removed. Instead, R defines the mapping from frame $\mathcal{B}$ to $\mathcal{I}$ and R_r the mapping from the reference frame $\mathcal{R}$ to $\mathcal{I}$. Differentiation of the translational tracking error coordinates results in the translational tracking error dynamics

$$\dot{\rho}_e = -S(\omega_r)\rho_e + \nu_e \tag{2.9a}$$

$$\dot{\nu}_e = -S(\omega_r)\nu_e + \frac{f}{m}R_r^\top R e_3 - \frac{f_r}{m}e_3. \tag{2.9b}$$

From (2.9b) a virtual input is defined as

$$u = \frac{f}{m}R_r^\top R e_3 - \frac{f_r}{m}e_3, \tag{2.10}$$

to obtain

$$\dot{\rho}_e = -S(\omega_r)\rho_e + \nu_e \tag{2.11a}$$

$$\dot{\nu}_e = -S(\omega_r)\nu_e + u. \tag{2.11b}$$

which is used to design tracking control laws. The feedback

$$u = -\sigma(k_\rho\rho_e + k_\nu\nu_e), \tag{2.12}$$

designed by [2] was proven to be uniformly globally asymptotically stable and uniformly locally exponentially stable. For these definitions, refer to [24]. Furthermore, note the presence of σ, which is a saturation function as defined in Section 2.4 in Definition 2.4.1.

From [2], also tracking attitude error coordinates and dynamics can be obtained, together with the proposed input, τ, to stabilize the tracking attitude dynamics.

Consider the dynamics (2.4c)-(2.4d) and their equivalents for the reference attitude dynamics. Define the errors $R_e = R_r R^\top$, $\tilde{R} = \hat{R} R^\top$, $\omega_e = \omega_r - \omega$, and $\tilde{\omega} = \hat{\omega} - \omega$, where $\hat{x}$ defines the estimated value of vector x resulting from an observer. Then the input

$$\tau = \tau_r + S(J\hat{\omega}_e)\omega_r + K_\omega \hat{\omega}_e + \sum_{i=1}^{n} k_i S(R_r^\top v_i)\hat{R}^\top v_i \tag{2.13}$$

$$\dot{\hat{R}} = \hat{R} S(\omega + \delta_R) \tag{2.14a}$$

$$J\dot{\hat{\omega}} = S(J\omega)\omega + \tau + \delta_\omega, \tag{2.14b}$$

where the innovation terms δ_R and δ_ω are given by

$$\delta_R = -c_R \sum_{i=1}^{n} k_i S(\hat{R}^\top v_i)(R_r^\top v_i + R^\top v_i) \tag{2.15a}$$

$$\delta_\omega = -c_\omega J S(\omega_r)\omega_e - c_\omega K_\omega \omega_e - C_\omega \tilde{\omega}, \tag{2.15b}$$

with $K_\omega = K_\omega^\top > 0$, $C_\omega = C_\omega^\top > 0$, $c_R > 0$, $c_\omega > 0$, and $k_i > 0$ such that $M = \sum_{i=1}^{n} k_i v_i v_i^\top$ has distinct eigenvalues, renders the equilibrium point $(R_e, \tilde{R}, \omega_e, \tilde{\omega}) = (I, I, 0, 0)$ UaGAS and ULES.

Now, Lefeber, van den Eijnden, and Nijmeijer (2017) noted that by rewriting (2.10) the aim could be to use f and τ to let $f R_r^\top R e_3$ converge to the vector $f_r e_3 + mu$. A fitting thrust could therefore be defined as

$$f = ||f_r e_3 + mu||, \tag{2.16}$$

which also ensures $f > 0$. Then, by defining a new rotation matrix R_d, which maps the reference frame $\mathcal{R}$ to a desired frame $\mathcal{D}$, the goal to determine τ which makes $f R_r^\top R e_3$ converge to the vector $f_r e_3 + mu$ can be replaced with the goal to determine a τ which makes $R_r^\top R$ converge to R_d. By proposing $R_e = R_r R_d R^\top$, $\omega_e = R_d^\top \omega_r + \omega_d - \omega$ and $\hat{\omega}_e = R_d^\top \omega_r + \omega_d - \hat{\omega}$, then the equilibrium point $(\rho_e, \nu_e, \tilde{\rho}_e, \tilde{\nu}_e, \tilde{z}, R_e, \tilde{\omega}, \tilde{R}, \omega_e) = (0, 0, 0, 0, 0, I, 0, I, 0)$ is UaGAS and ULES. Note that $\tilde{z}$ defines a term used for filtering observed position tracking errors, which is ignored further. More details on the desired frame $\mathcal{D}$, how it is defined and how it leads to R_d and ω_d, is provided later in this thesis.

2.3 Input-Output linearization

The concept of Input-Output linearization is used for this thesis. Input-Ouput linearization is a specific form of feedback linearization. Another form of feedback linearization includes full-state linearization. Both methods, discussed thoroughly in [24], propose a state feedback control to transform a nonlinear system. However, where full-state linearization linearizes the complete state equation without necessarily linearizing the output equation, Input-Output linearization specifically linearizes the input-output map while the state equation may be only partially linearized. For example, consider a class of nonlinear systems of the form

$$\dot{x} = f(x) + G(x)u \tag{2.17a}$$

$$y = h(x), \tag{2.17b}$$

where $\dot{x}$ described the derivative of x, which specifies the state space of a system. Also here, f and G are linear or nonlinear functions and u is the input variable. Feedback linearization poses the question of whether there exists a state feedback control

$$u = \alpha(x) + \beta(x)v, \tag{2.18}$$

where α and β are (non)linear functions and v the state feedback control, such that for full-state linearization, using a change of variables of the form

$$z = T(x), \tag{2.19}$$

transforms the nonlinear system into an identical system, but linear. This transformation can, oppositely, also cause the output to be transformed into a nonlinear term. Input-Output linearization however, specifies a control input u, of the form (2.18), such that

$$\dot{x} = v \tag{2.20a}$$
$$y = h(x), \tag{2.20b}$$

where now a tracking control problem can be solved using linear control theory. Note that Input-Output linearization is only possible when the (sum of the) relative degree(s) of the output(s) is smaller than or equal to the dimension of the state and only linearizes the input-output map. The definition of relative degrees is presented in the next section in Definition 2.4.2. Khalil explains the linearization of only the input-output map by the following example. Here, a full system is described as

$$\begin{aligned}\dot{x}_1 &= a\sin(x_2)\\ \dot{x}_2 &= v\\ y &= x_2.\end{aligned}$$

Clearly, the relation between input and specified output is purely linear. It is also apparent that state variable x_1 is not connected to the output, while still being highly nonlinear. This has to be accounted for when using Input-Output linearization, because these *zero dynamics* can result in unstable behaviour.

If a system (2.17) is decomposed in an external part, ξ, and an internal part, η, it is said to be in *nominal form*. This external part can be linearized by the state feedback control (2.18), while the internal part is made unobservable by the same control. If the external part is set to be 0, the internal dynamics can be rewritten to

$$\dot{\eta} = f(\eta, 0), \tag{2.21}$$

which is called the *zero dynamics*. If the system (2.21) has an asymptotically stable equilibrium point in the domain of interest, the system is said to be *minimum phase*.

2.4 Definitions

The following section provides additional definitions, theorems and mathematical methods that are used throughout this thesis.

Saturation functions

Consider $\sigma(s(\cdot)) : \mathbb{R}^n \to \mathbb{R}^n$ to be a vector-function which is twice continuously differentiable and monotone and where $s(\cdot)$ satisfies $s(0) = 0$ and $\lim_{x\to 0} s(x)/x = s'(0) > 0$. Also, let $V_\sigma(e) = \int_0^{e^\top e} s(x)/x\mathrm{d}x$, which is positive definite and radially unbounded.

Definition 2.4.1 (cf. [1, Definition 1]). *A function σ_i for which $||\sigma_i(e)|| \leq \gamma$ for all e is called a saturation function.*

Examples for saturation functions are $\sigma(x) = \gamma\frac{x}{\sqrt{1+x^\top x}}$ or $\sigma(x) = \gamma\tanh(||x||_2/\gamma)||x||_2^{-1}$, with a saturation bound $\gamma > 0$ that either decreases the available domain for small values, or increases the available domain for large values.

Flatness

A system is flat if it is possible to find a set of outputs, which is equal to the number of inputs, such that all states and inputs can be determined from these outputs without integration [25]. For the exact definition of flatness, one is referred to [16].

Furthermore, if a reference to a flat output is made, it is meant that reference trajectories are specified from these flat outputs without the need of integration, provided they are sufficiently smooth. The smoothness of a function is a property that defines the number of derivatives in a certain domain. Sufficiently smooth therefore indicates that the function should be differentiable at least as many times needed to define the reference trajectories.

Lie derivative [24]

Consider the following single-input-single-output (SISO) system

$$\dot{x} = f(x) + g(x)u \tag{2.22a}$$
$$\dot{y} = h(x), \tag{2.22b}$$

where the sufficiently smooth functions f, g and h lie in a domain $D \subset \mathbb{R}^n$. The derivative $\dot{y}$ is given by

$$\dot{y} = \frac{\partial h}{\partial x}[f(x) + g(x)u] \doteq L_f h(x) + L_g h(x)u, \tag{2.23}$$

where

$$L_f h(x) = \frac{\partial h}{\partial x} f(x), \tag{2.24}$$

is called the *Lie Derivative* of h with respect to f. Using this notation is convenient, since it is concisely written when calculating the derivative is repeated with respect to the same vector field or another new one. Some notations can be found below

$$\begin{aligned} L_g L_f h(x) &= \frac{\partial (L_f h)}{\partial x} g(x) \\ L_f^2 h(x) = L_f L_f h(x) &= \frac{\partial (L_f h)}{\partial x} f(x) \\ L_f^k h(x) = L_f L_f^{k-1} h(x) &= \frac{\partial (L_f^{k-1} h)}{\partial x} f(x) \\ L_f^0 h(x) &= h(x) \end{aligned}$$

Relative degrees

Definition 2.4.2 (cf. [24, Definition 13.2]). *A nonlinear system,* (2.17)*, has relative degree* k*,* $1 \leq k \leq n$ *in a region* $\mathcal{D}_0 \subset \mathcal{D}$ *if*

$$L_G L_f^{i-1} h(x) = 0, \qquad i = 1, 2, \ldots, k-1; \qquad L_G L_f^{k-1} h(x) \neq 0 \tag{2.25}$$

for all $x \in \mathcal{D}_0$.

In the special case that $k = n$, the nominal form reduces to a system where no internal part η exists. In this case, the system has no zero dynamics and, by default, is said to be minimum phase and thus has no unobservable internal dynamics.

Error measure for $SO(3)$

From [2], an error measure is defined to compare elements of $SO(3)$. Hereto, define a measure by its associated logarithmic map log: $SO(3) \to \mathfrak{so}(3)$, as

$$d(R_1, R_2) = ||\log(R_1, R_2^\top)|| \in [0, \pi]. \tag{2.26}$$

In other words, this error defines the absolute distance between two orientations on a sphere.

Chapter 3

Position Control

As mentioned in Section 1.4, this research aims to improve and extend on earlier work from [1] and [2] since unwanted tracking performance was observed when using these control laws. In the prior, a cascade control structure has been identified which led to control laws capable of sufficient tracking through corrupted measurements and saturation effects. Lefeber, Greiff, and Robertsson (2020) extended this work by designing a filtered output feedback controller with proven uniform local exponential stability, without relying on full state information. However, at the core of both these controllers lies the cascaded control structure and in both papers control laws are designed for the position control first, subsequently for the attitude control and finally for the cascaded closed-loop system.

This thesis follows the same cascaded structure and the same sequence for designing control laws, therefore firstly, in this chapter, the position control subsystem is analyzed. This is necessary since Section 1.3 has shown undesirable behaviour in the complete closed-loop system. More specifically, Figure 1.3 showed undesirable transient translations in that it already rotated without being near to the reference position, and Figure 1.4 showed unnecessary altitude changes, since the drone initially had the correct altitude. As both observations could be related to the previously designed position control law, this chapter provides more insight on how this position control law influences the closed-loop dynamics. First, however, Section 3.1 explains in more detail the motivation for analyzing the position control law after which, in Section 3.2, the position control law is validated. This chapter ends by giving some concluding remarks in Section 3.3.

3.1 Motivation

The motivation to start analyzing the position control design comes from initial testing of the complete system. As explained in Section 1.3, relatively simple trajectories still caused the quadrotor to either gain or lose altitude at the start of its movements without having initial altitude errors. Relatively more aggressive trajectories also had undesired transient behaviour, which is clearly visible in Figure 1.3b. A possible reason for these errors in transient behaviour was found in the translational tracking dynamics. By proposing a trajectory where the drone has a time dependent position and velocity reference and where the initial conditions were set such that

$$\rho_e(t_0) = \begin{bmatrix} \rho_{e,1} & \rho_{e,2} & 0 \end{bmatrix}^\top \qquad \nu_e(t_0) = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^\top \qquad R(t_0) = R_r, \tag{3.1}$$

then, from (2.11a), it can be seen that initial errors in the horizontal plane can already result in tracking errors in the e_3-direction without initially having errors in this direction. The equations

(3.2) show where those errors originate from

$$\dot{\rho}_e = -\begin{bmatrix} 0 & -\omega_{r,3} & \omega_{r,2} \\ \omega_{r,3} & 0 & -\omega_{r,1} \\ -\omega_{r,2} & \omega_{r,1} & 0 \end{bmatrix} \rho_e + \nu_e \tag{3.2a}$$

$$\dot{\rho}_e e_3 = \begin{pmatrix} -\omega_{r,2} & \omega_{r,1} & 0 \end{pmatrix} \begin{pmatrix} \rho_{e,1} \\ \rho_{e,2} \\ 0 \end{pmatrix} + 0. \tag{3.2b}$$

And since ρ_e is used for the feedback output (2.12), this could result in unnecessary initial increases or decreases in altitude during transient behaviour. This research aims to improve the transient behaviour of the position by designing a control law which enables the drone to fly in a straight path towards its target, therefore it makes sense to start validating the position control first.

3.2 Validation

In order to validate the position tracking control law from [1] and [2], it is best to recall the desired behaviour which is to be validated. As stated in Chapter 1, this research aims to design a control law which improves transient behaviour by flying in a direct path to its target. Thus, it would be desired if the drone would move in a straight line to its target.

To validate if the current position tracking control satisfies this behaviour, first, control laws which achieve straight line tracking of the position are defined. These control laws are ordered such that each consecutive control law is more realistic. Hereto, first a control law is designed that ensures movement along a straight line, without involving orientation in any way and finally, a control law is designed which is capable of tracking a time dependent orientation as well. Using this final controller and comparing it to the previously designed control law, this thesis tries to draw conclusions on the suitability of the control law from [1].

3.2.1 Translational tracking control 1

Without the presence of drone attitude, the problem is defined as follows. From an initial position ρ_0 with velocity $\dot{\rho}_0$, the drone should follow a trajectory towards ρ_r with $\dot{\rho}_r$, such that there can only be movement along the line crossing through ρ_0 and ρ_r. This problem is discussed in more detail by using an example.

The drone, set at position ρ_0 at time $t = 0$, is hovering and should move to position ρ_r where it should hover as well. Therefore:

$$\rho(0) = \rho_0 \neq \rho_r \qquad\qquad \dot{\rho}_0 = \dot{\rho}_r = 0 \tag{3.3}$$

where the dynamics are defined as

$$\ddot{\rho} = u, \tag{3.4}$$

in which u is the control input. Movement along a straight line through points ρ_0 and ρ_r is defined as

$$\begin{aligned} \rho(t) &= \alpha(t)\rho_0 + [1 - \alpha(t)]\,\rho_r \\ &= \alpha(t)\,[\rho_0 - \rho_r] + \rho_r, \end{aligned} \tag{3.5}$$

where α is some time dependent function scaling the current position to the distance ρ_0 and ρ_r. By differentiating (3.5) two times, the desired dynamics can be obtained.

$$\dot{\rho}(t) = \dot{\alpha}(t)\,[\rho_0 - \rho_r] \tag{3.6a}$$

$$\ddot{\rho}(t) = \ddot{\alpha}(t)\,[\rho_0 - \rho_r] = u. \tag{3.6b}$$

What can be noticed from (3.6) is the absence of feedback terms, making it an open-loop (feed-forward) controller. By adding feedback terms, therefore making it a closed-loop controller, the system improves in stability and robustness without losing the advantages of feed-forward.

Define

$$e = \rho - \rho_r, \tag{3.7}$$

which leads to the corresponding error dynamics

$$\ddot{e} + k_d\dot{e} + k_p e = 0, \tag{3.8}$$

with $k_p > 0$, $k_d > 0$, such that the system is Hurwitz [26]. Rewriting (3.8) to

$$\ddot{e} = -k_d\dot{e} - k_p e \tag{3.9}$$

results in stable translational error dynamics. Note however, that to this point no input is defined that is similar to (3.4). Therefore, combine the definition in (3.7) with the error dynamics of (3.8), to obtain

$$\ddot{\rho} - \ddot{\rho}_r = -k_d\dot{e} - k_p e, \tag{3.10}$$

which can be rewritten to

$$\ddot{\rho} = \ddot{\rho}_r - k_d\dot{e} - k_p e = u, \tag{3.11}$$

to obtain the input u that ensures straight line translational tracking dynamics.

3.2.2 Translational tracking 2

The general idea for designing a control law that includes time dependent attitude references is similar to the previous method. Again, the goal is to design a controller as in (3.11). This time however, R_r enters the system. Therefore it has to be accounted for and since R_r is time dependent, there has to be accounted for the derivatives of R_r as well. As can be seen in (2.4c), ω_r terms arise through the first derivative. Similarly, τ_r arises through the second derivative of R_r, which on its turn originates from (2.4d). It is therefore not straightforward to implement the exact same methods as in the previous subsection.

Now, instead of using the error coordinates as defined in (3.7), define

$$\rho_e = R_r^\top(\rho_r - \rho) \tag{3.12a}$$

$$\nu_e = R_r^\top(\dot{\rho}_r - \dot{\rho}) = \nu_r - R_r^\top R\nu. \tag{3.12b}$$

From (3.12a), it can be noted that the definition for the position tracking error is not described by $\rho - \rho_r$, which is often used in literature, for instance in [27]–[29]. Instead, the same position tracking error as in [1] is used. One of the reasons for using their definition is that it is independent of the definition of the inertial frame, since it is defined in the tracking reference frame $\mathcal{R}$, but the main reason comes from the resulting combination of input thrust and orientation when deriving the error dynamics.

Using these error coordinates, the following error dynamics can be obtained

$$\begin{aligned}\dot{\rho}_e &= \dot{R}_r^\top(\rho_r - \rho) + R_r^\top(\dot{\rho}_r - \dot{\rho}) \\ &= -S(\omega_r)R_r^\top(\rho_r - \rho) + R_r^\top(R_r\nu_r - R\nu) = -S(\omega_r)\rho_e + \nu_e \end{aligned} \tag{3.13a}$$

$$\begin{aligned}\dot{\nu}_e &= \dot{R}_r^\top(\dot{\rho}_r - \dot{\rho}) + R_r^\top(\ddot{\rho}_r - \ddot{\rho}) \\ &= -S(\omega_r)R_r^\top(\dot{\rho}_r - \dot{\rho}) + R_r^\top(\ddot{\rho}_r - \ddot{\rho}) = -S(\omega_r)\nu_e + R_r^\top(\ddot{\rho}_r - \ddot{\rho}). \end{aligned} \tag{3.13b}$$

From the last term it is possible to replace $(\ddot{\rho}_r - \ddot{\rho})$ with the previously mentioned error dynamics in (3.10) and to combine it with the error coordinates from (3.12)

$$\begin{aligned}\dot{\nu}_e &= -S(\omega_r)\nu_e + R_r^\top\left[k_d(\dot{\rho} - \dot{\rho}_r) + k_p(\rho - \rho_r)\right] \\ &= -S(\omega_r)\nu_e - k_d\nu_e - k_p\rho_e \\ &= -S(\omega_r)\nu_e + u. \end{aligned} \tag{3.14}$$

then, an input $u = \ddot{\rho}_r - \ddot{\rho}$ can be recognized, which originates from the desired Hurwitz error dynamics in (3.10). Rewriting $u = \ddot{\rho}_r - \ddot{\rho}$, leads to

$$u = (f/m)R_r^\top Re_3 - (f_r/m)e_3, \tag{3.15}$$

from simple differentiation of the dynamics (2.4). This is also the foundation of the position tracking control law from Lefeber, van den Eijnden, and Nijmeijer (2017), where they assumed u to be a virtual input to achieve the desired position tracking error dynamics. Clearly, the attitude R cannot be an input as it is a result of the input torque, therefore the term virtual input is used.

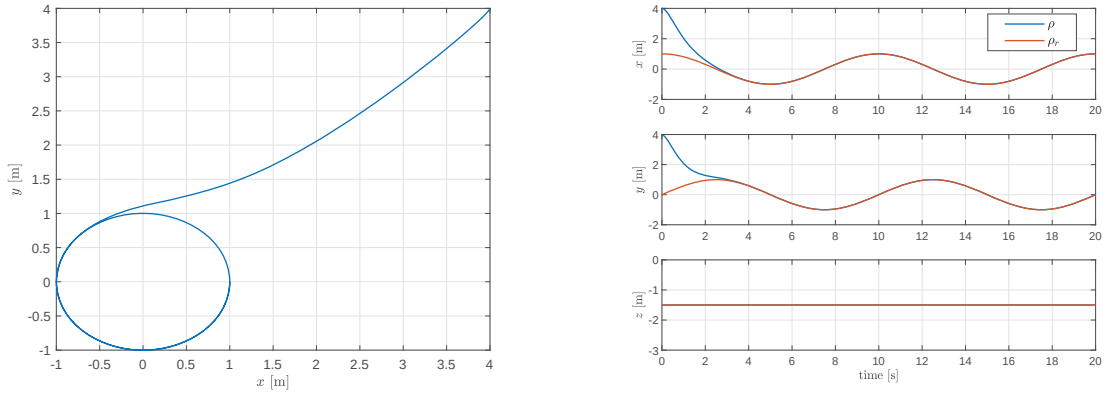
Lefeber, Greiff, and Robertsson (2020) proved that these closed-loop dynamics are UGAS and ULES and from (3.10) and (3.11) it can be concluded that the control input u tracks a trajectory which moves along a straight line towards ρ_r. Therefore, it can be concluded that the position control is not at fault for the behaviour described in Section 3.1. This is furthermore illustrated by the simulation in Figure 3.1, where

$$\rho_r = \begin{bmatrix}\cos(\omega_v) & \sin(\omega_v) & -1.5\end{bmatrix}^\top, \tag{3.16}$$

and

$$\rho_0 = \begin{bmatrix}4 & 4 & -1.5\end{bmatrix}^\top \qquad \nu_0 = \begin{bmatrix}0 & 0 & 0\end{bmatrix}^\top \qquad R_0 = R_r = I \qquad \omega_v = 0.2\pi \qquad k_p = k_d = 4,$$

which, by using (2.4a), also leads to ν_r. From Figure 3.1b, it can be observed that there is no initial



(a) *Quadrotor position in 2D frame direction.* **(b)** *Quadrotor positions and reference positions.*

Figure 3.1: *Resulting quadrotor position using* (3.14)

position error in the z direction and that by using the translational tracking control law from (3.14), it can also never appear. It can also be noted from 3.1a, that it is difficult to confirm if the drone translates in a straight line to the target, as the target is time dependent. Both observations however, indicate that the undesirable transient translations and altitude changes do not originate from the position control.

3.3 Concluding remarks

In this chapter a position tracking control law has been validated. The reason for validating this position tracking control law originates from unwanted results found using that control law. Instead of directly validating the entire system, the choice was made to design a control law which does not account for attitude dynamics in any way, which would behave according to the goals stated in Chapter 1. The final controller designed in this chapter could track time dependent position references and time dependent attitude references. While designing this final controller, described in (3.14), it becomes clear that it is equivalent to the controller designed in [1] and [2]. Figure 3.1, confirmed that the translational tracking control law does not create altitude errors if there are initially

none and it also showed that the transient translations from Figure 1.3a are not a direct result from the translational tracking control law. Therefore it can be concluded that the unwanted behaviour cannot be caused by the position control law.

In Chapter 4 different attitude control laws are validated, to examine if they could lead to undesirable behaviour.

Chapter 4

Attitude control

In the previous chapter, it has been shown that the unwanted altitude changes and the transient translations explained in Section 1.3 are not the result of the position tracking controller from [2]. Instead, it performs as desired, which means that the root of the undesirable behaviour is still unknown. Therefore, the next step is to validate the attitude tracking controllers from [1] and [2]. Designing a controller which can track a path in a straight line means not only designing a fitting position controller, it also means designing a controller which rotates such that the desired horizontal translations can be achieved and a controller that ensures the desired altitude translations. The latter is discussed later in this report, the former is discussed in this chapter. The attitude control laws from [1] and [2] are both capable of tracking "easy-to-follow" trajectories and trajectories involving aggressive maneuvering, but do they also cause the unwanted transient translations or altitude changes or both? Section 4.1 is dedicated to answering this question by simulating and presenting the attitude of the drone using the attitude controller from [2], since the attitude controller from [1] includes unnecessary terms which are left out in [2]. In Section 4.2 the possibility of designing a new attitude tracking controller is discussed, where the main focus lies on a controller based on using a feedback linearization technique known as "Input-Output linearization". Finally, in Section 4.4 some concluding remarks are given.

4.1 Motivation

Both [1] and [2] have proposed a method for tracking attitude dynamics. Similar methods were used, in that they both define error measures for R and ω which they use to define a stabilizing control input for τ, however each has their own version of the final control input. Again, both these versions are proven to be ULES and UaGAS. Contrary to the position control however, it is unclear how these control inputs exactly rotate in order to stabilize the system. It is clear from the simulations shown in [1] and [2] and independent simulations, using either method, similar initial conditions and similar trajectories, that settling times are relatively quick, but whether these rotations are fitting for the goal of this report is to be decided.

In order to conclude if either method could be applicable for this research, attitude dynamic simulations were conducted to visualize the attitude for multiple initial orientations. As mentioned, only simulations using the attitude controller by [2] are performed. This for the reasons that the attitude controller from [1] includes unnecessary nonlinear terms and because these simulations can be performed simply by using the control law from [2] and assuming there are no observer errors in (2.13). By solely focusing on the attitude it is not difficult to imagine what the desired attitude dynamics should be. Therefore, the attitude dynamics and complementary controllers are extracted from the system and evaluated separately.

Starting by simulating the first situation. Here, the reference orientation is set to be

$$R_r = I, \tag{4.1}$$

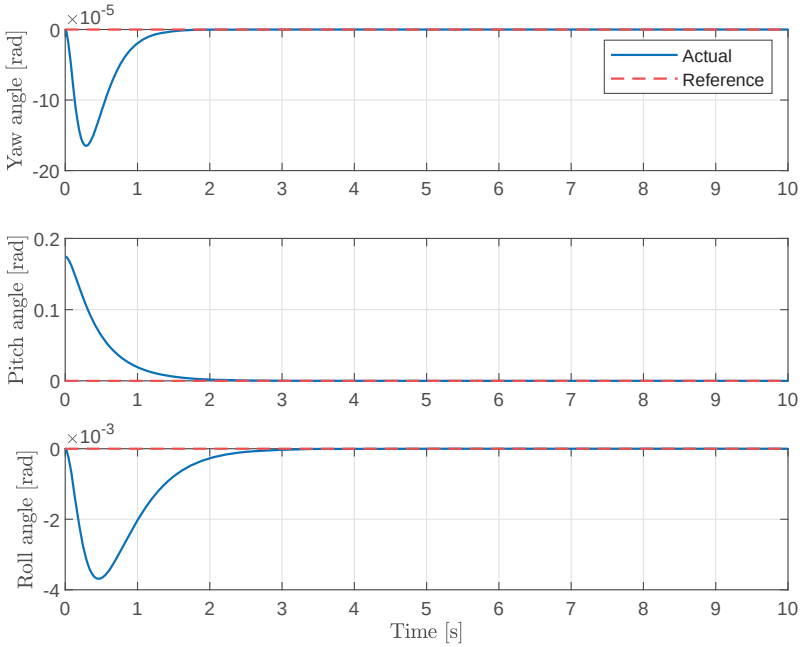


Figure 4.1: *Attitude and reference in Euler angles from controller by [2].*

which indicates a hovering position. The initial orientation is set as

$$R(t_0) = \begin{bmatrix} 0.9848 & 0 & 0.1736 \\ 0 & 1 & 0 \\ -0.1736 & 0 & 0.9848 \end{bmatrix},$$

which occurs after a pitch, θ, rotation of 10 degrees. Under ideal circumstances, the quadrotor would rotate with negative pitch angles to rotate back to the reference attitude. In no situation it would be necessary to rotate using either roll or yaw, knowing that this would not benefit in moving along a straight line. See Figure 2.1 for the corresponding angle representations.

Results can be seen in Figure 4.1. From here, it is visible that the system converges quickly. However, what can also be noticed is that both roll and yaw rotations occur. Even though these rotations are small, they are large enough to cause movements in undesired directions. Only by rotating directly to a reference position it is possible to translate in a straight path towards that reference position. Therefore, the result obtained by using the attitude controller from [2] does not suffice and alternative methods are thus necessary.

4.2 Alternative attitude control

Before an alternative attitude controller can be designed, it is best to start defining the requirements of this controller. Most importantly, the new controller should be able to rotate towards its reference orientation as fast as possible, without using rotations which do not contribute to the goal of the position control. Next to that, only rotations around the body-fixed x and y axis can lead to the reference orientation, since the goal of this thesis is to control the heading direction, which follows from rotations around the body-fixed z axis, independently too. In the next part of this section, the method described as Input-Ouput linearization is proposed as solution.

4.2.1 Input-Output linearization

From Chapter 2, it is known that Input-Output linearization is a method that linearizes the input-output map of a system. This linear input-output map can then be used to solve a tracking control problem. If the input-output map of the quadrotor attitude dynamics can be linearized, it can be used to track attitude dynamics as specified. A similar approach is used in [27], however this research did not use cascade control structure, but a control structure with an inner and outer loop. In order to implement Input-Output linearization for the attitude dynamics described in (2.4c)-(2.4d), firstly the state space variables are specified

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} R(\phi, \theta, \psi) \\ \omega \end{bmatrix}. \tag{4.2}$$

Note that R is a result of sequential roll, pitch and yaw rotations defined by respectively ϕ, θ and ψ, as seen in (2.1). Next, the following outputs are proposed

$$y = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} \psi \\ \theta \\ \phi \end{bmatrix}. \tag{4.3}$$

Setting y to describe the roll, pitch and yaw angles ensures total control over the attitude of the quadrotor, if and only if the sum of the relative degrees k_i, of each output y_i, with $i = \{1, 2, 3\}$, is smaller than or equal to the dimension, n, of the state space x [30]. The relative degree of a system is defined by [24] and is discussed in Section 2.4.

After calculating the Lie derivatives two times for each output, it can be found that $L_G L_f h(x) \neq 0$, which concludes that the attitude dynamics are Input-Output linearizable. Furthermore, it holds that the sum of the relative degrees is exactly equal to the dimension, n, of the state space, x, in (4.2), which means there are no zero-dynamics. Appendix A provides a detailed description on how these results are obtained.

Knowing that y_i is Input-Output linearizable, what remains is to define τ such that the input-output map reduces to $y_i = v_i$. The following state feedback control is proposed

$$\tau = -S(J\omega)\omega + J \begin{bmatrix} -s\theta & 0 & 1 \\ c\theta s\phi & c\phi & 0 \\ c\theta c\phi & -s\phi & 0 \end{bmatrix} \left[v - M(\dot{\psi}, \dot{\theta}, \dot{\phi}, \omega)\omega \right], \tag{4.4}$$

which results in the desired, linear, input-output map. Note that s and c are abbreviations of sin and cos.

By achieving a linear input-output map, it remains to define the desired tracking behaviour. As mentioned in Section 3.2.1, a tracking control law to ensure linear movement can be described by (3.9), with in a like manner $k_{p,r} > 0$, $k_{d,r} > 0$ such that a Hurwitz system is ensured. Note that (3.7) cannot be used here, therefore instead let

$$e_i = y_i - y_{i,r} \tag{4.5a}$$
$$\dot{e}_i = \dot{y}_i - \dot{y}_{i,r}, \tag{4.5b}$$

where y_i defines ψ, θ and ϕ for $i = \{1, 2, 3\}$ respectively, from (4.3). Similarly, $\dot{y}_i$ defines $\dot{\psi}$, $\dot{\theta}$ and $\dot{\phi}$ for $i = \{1, 2, 3\}$ respectively.

In addition to the controller defined in Section 3.2.1, for the attitude error tracking dynamics, also a feedforward term has been implemented. This feedforward term can also be found in (3.11), but is used here as follows:

$$\ddot{e}_i = -k_{p,r} e - k_{d,r} \dot{e} \tag{4.6a}$$
$$\ddot{y}_i - \ddot{y}_{i,r} = -k_{p,r} e_i - k_{d,r} \dot{e}_i \tag{4.6b}$$
$$\ddot{y}_i = \ddot{y}_{i,r} - k_{p,r} e_i - k_{d,r} \dot{e}_i, \tag{4.6c}$$

where $\ddot{y}_i$ defines the second derivatives of (4.3).

Note that y_i can be derived from the rotation matrix described in (2.1), according to the pseudo code in Appendix A. Also, $\dot{y}_i$ can be derived from the expression of ω as function of y_i and $\dot{y}_i$ in [31], which can be rewritten to an expression of $\dot{y}_i$ as function of y_i and ω. The exact expression can be found in Appendix A, (A.2). Recall that for the sake of clarity Euler angles are used, Appendix A also presents the implemented method in $SO(3)$.

Considering the desire to achieve specific orientations by rotating in roll and pitch angles only, with the intend that yaw remains free to choose, the following approach from [1] is followed.

Define

$$f_d = \begin{bmatrix} f_{d1} \\ f_{d2} \\ f_{d3} \end{bmatrix} = \frac{f_r e_3 + mu}{||f_r e_3 + mu||}, \tag{4.7}$$

as the desired thrust direction, specified in more depth in Chapter 5. Then, let

$$R_d = \begin{bmatrix} 1 - \frac{f_{d1}^2}{1+f_{d3}} & -\frac{f_{d1}f_{d2}}{1+f_{d3}} & f_{d1} \\ -\frac{f_{d1}f_{d2}}{1+f_{d3}} & 1 - \frac{f_{d2}^2}{1+f_{d3}} & f_{d2} \\ -f_{d1} & -f_{d2} & f_{d3} \end{bmatrix}, \tag{4.8}$$

denote the desired rotation matrix which rotates the desired thrust vector, f_d, to the thrust vector of the reference (i.e. e_3) in the plane spanned by both vectors. By differentiating R_d and premultiplying it with R_d according to the dynamics in (2.4c), to obtain a definition for ω_d results in

$$\omega_d = \begin{bmatrix} -\dot{f}_{d1} + \frac{f_{d2}\dot{f}_{d1}}{1+f_{d3}} \\ \dot{f}_{d1} - \frac{f_{d1}\dot{f}_{d3}}{1+f_{d3}} \\ \frac{f_{d2}\dot{f}_{d1} - f_{d1}\dot{f}_{d2}}{1+f_{d3}} \end{bmatrix}. \tag{4.9}$$

Now, instead of retrieving $y_{i,r}$ from the entries of matrix R_d in (4.8) in combination with the definition of the rotation matrix in (2.1), where R is defined by RPY-angles, take

$$\begin{aligned} R_{RP} &= \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix} \\ &= \begin{bmatrix} c\theta & s\phi s\theta & c\phi s\theta \\ 0 & c\phi & -s\phi \\ -s\theta & c\theta s\phi & c\phi c\theta \end{bmatrix}, \end{aligned} \tag{4.10}$$

to define a rotation matrix, R_{RP}, consisting of only roll and pitch rotations. This rotation matrix ensures rotations are only in x and y direction, leaving the yaw angle free to define later. From the definition of R_{RP} and the entries f_{d1}, f_{d2} and f_{d3} in rotation matrix R_d, it is possible to find $y_{2,r}$ and $y_{3,r}$. As mentioned in Section 2.1.2, it is assumed that $\cos\phi \geq 0$ to guarantee uniquely defined rotation matrices, and Appendix A provides a pseudo code that calculates the Euler angles accordingly. Furthermore, $\dot{y}_{2,r}$ and $\dot{y}_{3,r}$ can be obtained by differentiating R_{RP} and using the dynamics from (2.4c) to obtain a skew symmetric matrix $S(y_{1,r}, y_{2,r}, \dot{y}_{1,r}, \dot{y}_{2,r})$, which defines the desired angular velocities. Then, by combining the definition of a skew symmetric matrix (2.5) and the entries of (4.9), let

$$S_{3,2}(y_{1,r}, y_{2,r}, \dot{y}_{1,r}, \dot{y}_{2,r}) = \omega_{d,1} \tag{4.11a}$$

$$S_{1,3}(y_{1,r}, y_2, \dot{y}_{1,r}, \dot{y}_{2,r}) = \omega_{d,2}, \tag{4.11b}$$

which are two equations with two unknowns, which can be solved to find $\dot{y}_{2,r}$ and $\dot{y}_{3,r}$. Finally, the expressions for $\ddot{y}_{2,r}$ and $\ddot{y}_{3,r}$ result from differentiation of the expressions for $\dot{y}_{2,r}$ and $\dot{y}_{3,r}$.

Recall from Section 2.1.2, that by using the parametrization in Euler angles, singularities can arise [23]. These singularities can enter the control law from the tracking dynamics in (4.6c), when retrieving orientation representation in Euler angles. If the rotation matrix from which the Euler angles are retrieved describes a Gimbal Lock orientation, it is not possible to uniquely define the roll and pitch angles. Greiff (2017) proposes a method to avoid these singularities, by augmenting the quadrotor dynamics (2.4) to keep the pitch constant at its most recently feasible value, θ_f, when sufficiently close to the singularity. Thus, in order to define y_i sufficiently correct at every instant, let

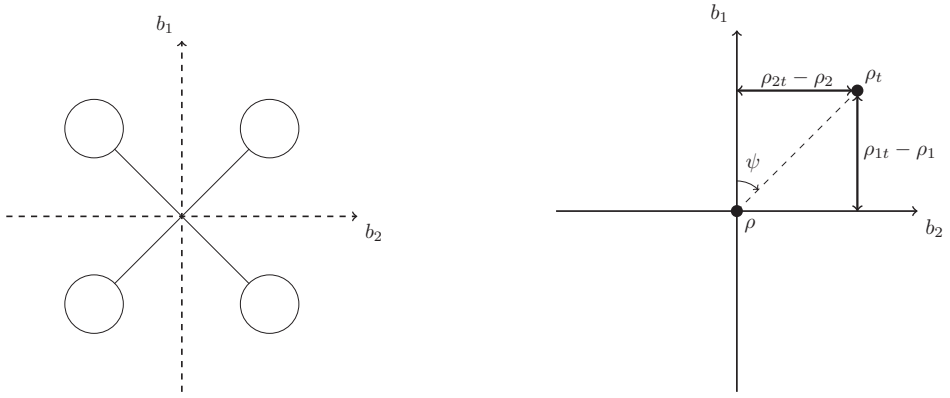
$$R = \begin{cases} R(\psi, \theta, \phi), & \text{if } ||\cos(\theta)|| > \epsilon \\ R(\psi, \theta_f, \phi), & \text{if } ||\cos(\theta)|| \leq \epsilon \end{cases}, \tag{4.12}$$

where ϵ defines a scalar bound. Now, to finalize the new attitude controller only the yaw reference angle needs to be defined.

4.3 Yaw angle reference

The previous section has provided a method to linearize the input-output map of the attitude dynamics to solve the tracking control problem of (4.6c) and it also provided methods to determine the outputs y_i, its derivatives and the reference outputs $y_{r,1}$, $y_{2,r}$ and its derivatives. The reference output $y_{3,r}$ and its derivatives have not been defined yet. This section provides the necessary expressions.

For this thesis, it is desired to define a yaw angle reference that is able to continuously face a specific reference target point, ρ_t, which lies in the horizontal plane spanned by $e_{1\mathcal{I}}$ and $e_{2\mathcal{I}}$. Since the drone itself is not necessarily idle in its global position, the yaw reference should adjust according to the changes in position. Figure 4.2 illustrates the correct yaw angle.



(a) *2D quadrotor view* **(b)** *Yaw calculation method*

Figure 4.2: *Yaw illustrations.*

The following yaw reference angle is proposed

$$\psi_r = \text{atan2}\left(\rho_{2t} - \rho_2, \rho_{1t} - \rho_1\right), \tag{4.13}$$

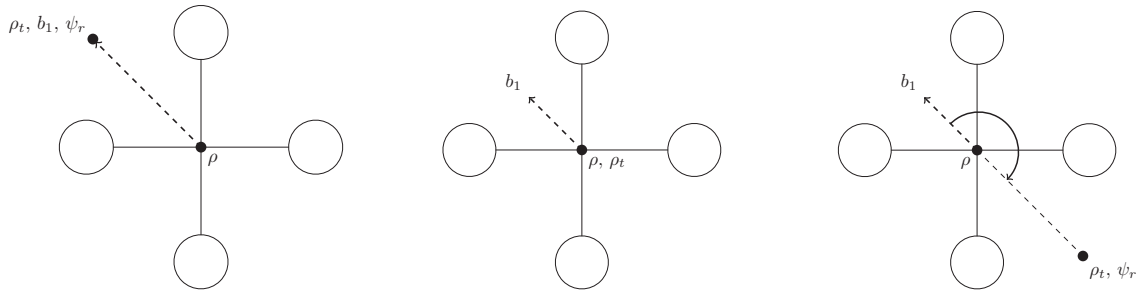
where atan2 is the 2-argument arctangent. The reference angular velocity $\dot{\psi}_r$ follows from the derivative of ψ_r

$$\dot{\psi}_r = \frac{-\dot{\rho}_1(\rho_{2t} - \rho_2)}{(\rho_{2t} - \rho_2)^2 + (\rho_{1t} - \rho_1)^2} - \frac{\dot{\rho}_2(\rho_{1t} - \rho_1)}{(\rho_{2t} - \rho_2)^2 + (\rho_{1t} - \rho_1)^2}, \tag{4.14}$$

and $\ddot{\psi}_r$ from differentiating $\dot{\psi}_r$.

It should be noted that singularities can occur using these formulations. For situations where $\rho_t = \rho$ the yaw reference is undefined. Furthermore, Figure 4.3 shows a situation where the drone is directly heading towards ρ_t. Before the drone has reached this reference target point, the yaw angle

and the desired yaw angle are equal thus resulting in no control action. As soon as $\rho_t = \rho$ and the yaw reference becomes undefined, the system can become unstable. Once the drone has passed this reference target point the yaw error is also immediately significant, which can lead to large control actions. Such undesirable situations can be avoided by using collision avoidance control laws, where the equilibria are asymptotically stable and no collisions between objects occur [32]. When using collision avoidance control laws, the drone moves around the reference target point so that it can never "collide" with it which also means that the yaw reference is continuously defined.



(a) $\psi_r = \psi = -1/4\pi$ **(b)** ψ_r *is undefined,* $\psi = -1/4\pi$. **(c)** $\psi_r = 3/4\pi$, $\psi = -1/4\pi$.

Figure 4.3: *Yaw illustrations.*

4.4 Concluding remarks

In this chapter the attitude tracking control problem has been considered. Even though the position tracking controller resulted in straight line movements, the quadrotor would still not achieve the requirements if the attitude tracking control is deficient. Previous control laws have been validated, which resulted in the conclusion that they did not suffice. This conclusion followed from the results obtained in Figure 4.1, which showed rotations along axis that would result in undesirable translations. Therefore, a new attitude tracking controller has been proposed.

This new attitude tracking controller is designed by using a feedback linearization method called, Input-Output linearization. The system (2.4c)-(2.4d) proved to be Input-Output linearizable, since the sum of the relative degrees of the outputs equaled the state dimension. It has been shown that the state feedback controller from (4.4) is able to linearize the input-output map. Therefore, what was left to do was to define the desired tracking behaviour. The final tracking control law is shown in (4.6c). In view of the goal to rotate the quadrotor around its x and y axis only to achieve the reference position, a method is proposed to define the suiting roll and pitch angles. This method uses the definitions from (4.10) and the values that result from (4.7) and (4.8).

The yaw reference has been specified separately. It has been designed such that it constantly faces a certain point in space, ρ_t, by defining the angle between the current position in the horizontal plane and the target point. The corresponding angular velocity and accelerations that match ψ_r have been derived through differentiation of (4.13).

The new attitude tracking control law allows for multiple possible scenarios of singularity. In case of singularities through gimbal lock, it has been shown that the method described by [31] in (4.12) can avoid these singular cases, by augmenting the attitude dynamics as in (4.12). However, in case of singularities induced by undefined yaw references, there has yet to be implemented a method that can prevent it. The method from [32] describes tracking control laws that include collision avoidance. It is suggested to implement similar control laws to define ρ_t as an object with which collisions do not occur so that the actual position ρ and the reference target point ρ_t are never equal and thus resulting in a continuously defined yaw reference.

This chapter includes a method to describe pitch and roll angles from the virtual input u, which is defined in Chapter 3, even though this method has not been discussed yet. The following chapter

therefore includes detailed derivations for the purpose of explaining where the roll and pitch angles originate from. Chapter 4 also discusses its effectiveness, to validate if the method remains suitable for this research or has to be altered.

Chapter 5

Combined control

From previous chapters it has become clear that the position control laws do not result in the unwanted altitude changes and it has become clear what the desired input torque, τ, should be in order to rotate in a manner that fits the goals of this research, namely to track a trajectory in the shortest possible way: in a straight line. Up until this point, little attention has been given to the other input of the total system, described in (2.4), being the thrust. Rotations alone do not guarantee straight line tracking, a fitting thrust per orientation is equally important. In Section 4.2 the variables f_d, R_d and ω_d were mentioned, without thoroughly explaining what they represent. There, attention has been given as to what their utility is for defining desired thrust directions and desired orientations. What has not been mentioned yet, is their convenience for defining the final thrust input.

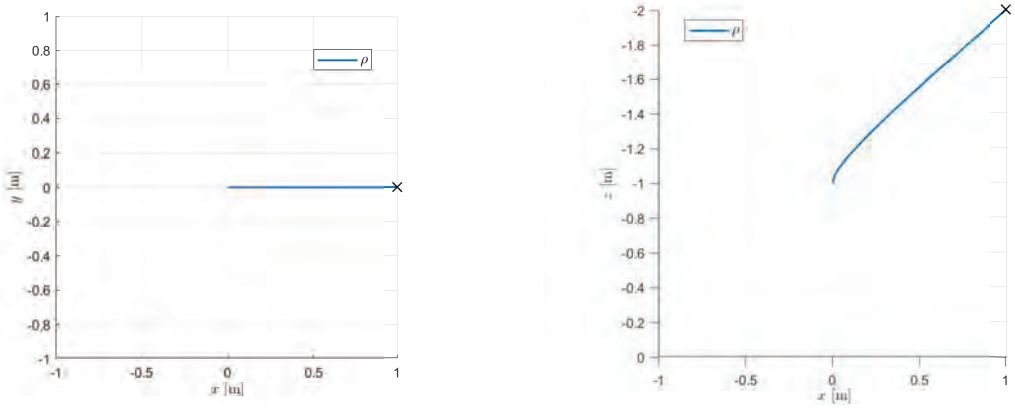
This chapter pays attention to deriving a thrust definition that increases tracking performance during transient altitude translations. As in earlier chapters, firstly a motivation is given regarding the previously defined thrust methods and their effectiveness. By means of illustrations these methods are validated. Next, two alternatives for the already available thrust definitions are proposed. The first is based on geometric properties of the vectors used to define the already existing thrust magnitude, the second is based on angular differences and corresponding scaling values. Both methods are described in depth and their respective strengths and flaws are identified. At last, some concluding remarks are given.

5.1 Motivation

As explained in Chapter 3, u defines a virtual input to stabilize the position tracking error dynamics. Being a virtual input, u on itself is not suitable to be implemented in the actual closed-loop system. The only inputs which are available are the input thrust, f, and the input torque, τ. In the previous chapter a definition for τ has been designed which would achieve the desired results of this research. It has been shown that this τ is capable of converging R to R_r such that no rotations occur which would lead to unnecessary translations. However, it is not yet possible to translate between two certain positions if the initial attitude is already equal to the reference attitude. This is why f_d, R_d and ω_d have been designed. They define the necessary thrust direction, attitude and angular velocity to achieve the desired translation, as explained earlier in Chapter 2. Lefeber, van den Eijnden, and Nijmeijer (2017) explained the reason behind using these equations is to replace the goal of determining a τ which converges $fR_r^\top Re_3$ to $f_r e_3 + mu$ to determine τ which converges $R_r^\top R$ to R_d, by using (2.16). Chapter 4 has shown a method to define the required roll and pitch angles from R_d to ensure straight line translations, but the suitability of f from (2.16) is yet to be determined.

As can be seen from Figure 1.4, this method induces altitude errors when translating in a horizontal plane while initially being at the correct altitude. It has to be noted that the exact method from [1] has been followed here and as Chapter 4 has shown, the attitude control was not suitable for this research. Therefore, in order to validate the thrust definition from (2.16), it has been implemented in a controller which consists of the validated position tracking controller for the position

error subdynamics, (3.11), and the newly defined attitude controller for the attitude subdynamics, (4.4), with tracking dynamics based on (3.9). Figure 5.1 shows the results. From here, it can be



(a) *Position tracking response in XY-plane.* **(b)** *Position tracking response in YZ-plane.*

Figure 5.1: *Translations using new position and attitude controllers, but with non validated thrust definition.*

seen that the altitude translations do not match the horizontal translations. Even before the attitude is changed enough for it to translate horizontally, the altitude increases already. After a certain amount of time the altitude translations match the horizontal translations, indicating that $R_r^\top R$ has converged to R_d. Nonetheless, since the goal of this research is to ensure straight line movements, the thrust definition from (2.16) is deemed unsuitable. Therefore, options for defining a new thrust equation have to be explored, which is the subject of the two following sections.

5.2 Geometric thrust solution

This section proposes an alternative for the thrust definition from (2.16). This proposal is based on finding a solution to the geometric problem that arises when using the desired rotation matrix R_d in combination with u, the orientation of the thrustvector, given by Re_3, and most importantly the thrust definition from (2.16). Figure 5.2 visualizes this exact problem. In this figure, one can see a

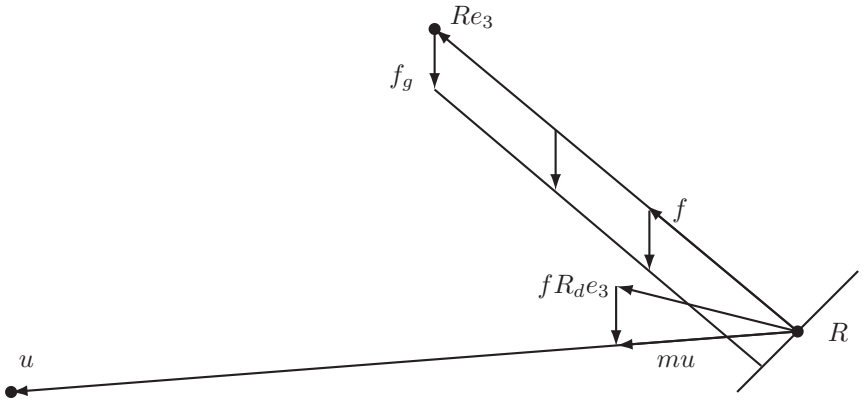


Figure 5.2: *Schematic visualization of thrust problem in two dimensions.*

two-dimensional representation of the attitude of a quadrotor, represented by the line annotated with R. Additionally, multiple vectors are shown, Re_3 which defines the direction of the current thrust, u which defines the desired resulting (negative) acceleration direction, obtained from (3.11), R_de_3 which defines the desired thrust direction and $\vec{f_g}$ which defines the gravitational force in opposite direction. Recall, that by using a NED-frame, e_3 is aimed towards the ground. Figure 5.2 visualizes

the behaviour that occurs when using the thrust definition from [1] and [2]. Here, one can see that the desired thrust direction, $R_d e_3$, is scaled with f from (2.16) to exactly match $f_r e_3 + mu = f R_d e_3$, where f_r defines the gravitational force, f_g. The resulting acceleration direction exactly matches the desired acceleration direction. However, clearly $R_d e_3 \neq Re_3$, which leads to a resulting acceleration direction that does not match mu, since f is too large for this situation. In Figure 5.2 a two-dimensional situation is shown, but it has to be noted that this problem is not limited to a two-dimensional space. By imagining that the vector Re_3 is rotated such that it now points inwards or outwards of the paper, the problem immediately becomes three-dimensional.

5.2.1 Solving for two-dimensional situation

Rather than solving this problem in three dimensions directly, it is firstly considered as a two-dimensional problem. For guaranteeing straight line translations, accelerations are needed in the direction of u, the scale of this vector solely defines the speed at which it would translate along the trajectory. Therefore, by formulating f, such that $f Re_3 = f_r e_3 + kmu$, where k defines some scalar value, straight line translations would be guaranteed. Figure 5.3 illustrates the resulting situation.

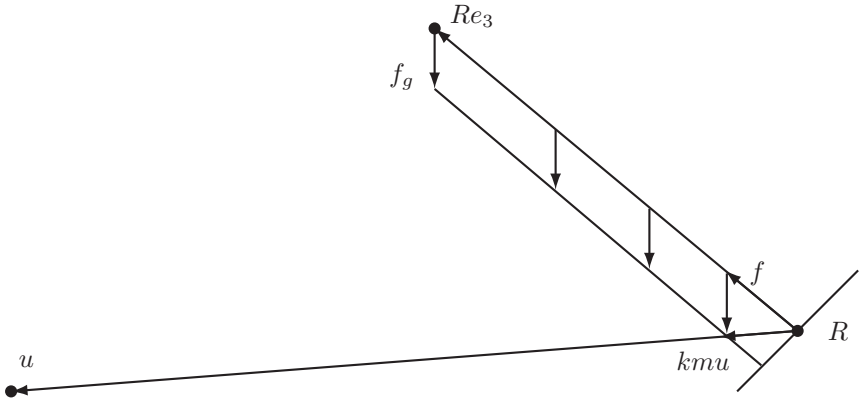


Figure 5.3: *Schematic visualization of the desired solution to the thrust problem in two dimensions.*

The resulting acceleration vector is evidently smaller than the one in Figure 5.2, since it is scaled with k, yet this acceleration direction is feasible. The quadrotor slowly, but accurately, follows the desired trajectory when using this thrust definition.

To determine the fitting thrust definition, this problem is solved by identifying it as a system with two linear equations, which by using the matrix equation can be written as

$$\mathbf{A}\vec{x} = \vec{b}, \tag{5.1}$$

where

$$\mathbf{A} = \begin{bmatrix} Re_3 & mu \end{bmatrix} = \begin{bmatrix} Re_{3,1} & mu_1 \\ Re_{3,2} & mu_2 \end{bmatrix}, \tag{5.2}$$

$$\vec{b} = \begin{bmatrix} f_{r,1} \\ f_{r,2} \end{bmatrix}, \tag{5.3}$$

$$\vec{x} = \begin{bmatrix} f \\ k \end{bmatrix}. \tag{5.4}$$

Since $\mathbf{A}$ is a n-by-m matrix that, in the two-dimensional situation, becomes a square matrix of size n, the equality can be solved by using

$$\vec{x} = \mathbf{A}^{-1}\vec{b}, \tag{5.5}$$

if the equations are independent, i.e. the matrix $\mathbf{A}$ has full rank. By identitfying $f = \vec{x}_1$, the geometrically ideal thrust definition can be found.

5.2.2 Solving for three-dimensional situation

For situations in 3 dimensions, this method is impracticable, since *exact* intersections between fRe_3 and $f_r e_3 + mu$ rarely occur. It is at this point too, that it has to be noted that straight line movements are not feasible for each initial attitude. For instance, if Re_3 lies in the plane spanned by e_1 and e_3 only, it is initially not possible to directly translate towards a point in e_2. Using the attitude tracking controller from (4.4) with (3.9) guarantees rotations directly towards e_2, but this is not immediate. Therefore, decisions have to be made as to what defines the desired behaviour.

A possible solution is to define a new frame, R_u, which rotates the reference frame R_r around its z axis such that u becomes a two-dimensional vector. By doing so, matrix equations are applicable once more if the second entry of Re_3 is ignored. As a result of this method, the quadrotor follows a straight line trajectory in the frame R_u. This does not guarantee that the quadrotor follows a straight line trajectory in the inertial frame, but as previously stated, this is not possible for some scenarios. In the figure below, a schematic representation can be seen of the rotation matrix R_u.

Figure 5.4: *Visualization of rotation matrix R_u.*

5.2.3 Flaws

Even though the previously determined method to define a geometrically ideal thrust seems suitable, multiple flaws cause the contrary. The first flaw is illustrated in Figure 5.5. According to this method, the geometrically ideal thrust finds itself on the intersection between fRe_3 and $f_r e_3 + kmu$. In the scenario described in Figure 5.5, this would lead to a thrust of $f = 0$ and $k = f_r e_3/mu$, which would result in the quadrotor falling straight down. As the absence of upwards thrust also causes the inability to generate torque, the quadrotor would not be able to rotate and therefore, would keep falling until it hits the ground. For this reason [1] and [2] constrained $0 < f_r^{\min} \leq f_r$ and therefore $f_d > 0$. Another flaw resulting from the method to define the thrust by using matrix equalities, is the possibility of negative thrusts. Figure 5.6 illustrates such a scenario. The desired acceleration of the quadrotor has positive terms in the e_2 direction, whereas the thrust direction has negative terms in the e_2 direction. By using (5.5) an intersection is calculated between fRe_3 and kmu. From Figure 5.6 it can be observed that there is one unique intersection point. However, the resulting thrust at this intersection point is negative, which is unwanted and even impossible to achieve for quadcopters as they only rotate in one direction. Again, for this reason [1] and [2] constrained $0 < f_r^{\min} \leq f_r$ and therefore $f_d > 0$.

Since there are no achievable thrusts that would lead to straight line behaviour, there again has to be decided what the next best solution could be. Two possible alternatives can be proposed. Firstly, the minimal possible thrust that still achieves positive thrusts per rotor or, secondly, a high thrust to achieve torques capable of rotating the quadrotor as fast as possible to the upward position. Both however, imply that discontinuities arise.

Discontinuities are the main reason this method is deemed unfit. One of the discontinuities that arise when using this method, can be traced back to the main purpose of using (5.5), being that it calculates scaling factors $\vec{x}$ so that an equality between two systems can be found. An equality

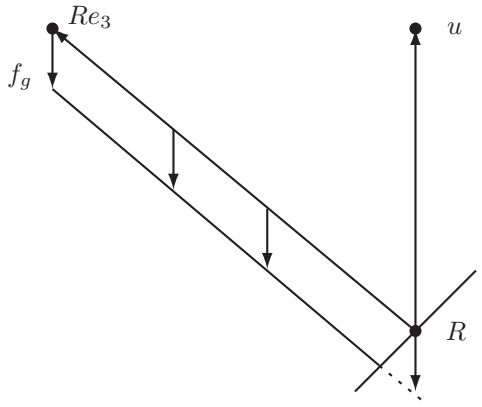


Figure 5.5: *Example 1, where $f = 0$ and the quadrotor would fall down.*

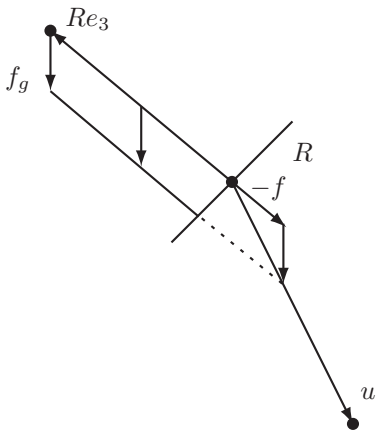


Figure 5.6: *Example 2, where $f < 0$ which is not achievable.*

between two systems in this case means an intersection point between fRe_3 and $f_r e_3 + mu$. In situations described in Figure 5.3, a clear intersection point can be derived. However, in situations visualized in Figure 5.7 no unique solution for f can be found.

Once the angle between Re_3 and mu, say γ, turns 0 or π so that $||Re_3|||| \cos(\gamma)|| = ||Re_3||$, there is no unique solution anymore. This is in line with the condition from (5.5), that the two systems need to be independent, thus guaranteeing $\mathbf{A}$ has full rank. Therefore, no unique solution can be found for f in such scenarios. From Lefeber, van den Eijnden, and Nijmeijer (2017), it is known that a fitting solution is defined by (2.16), but by implementing conditions to the thrust definition, making it a hybrid controller, the controller loses continuity. Since the goal of this thesis is to define a continuous control law, the presence of discontinuities are highly inconvenient and therefore to be avoided if possible.

5.3 Scaling with attitude error

It is from Figure 5.7, another alternative for defining the thrust is determined. In order to fly in a straight path, starting from a hovering position, towards a point in three-dimensional space directly above or below the current point in three-dimensional space, a thrust force of $f = ||f_r e_3 + mu||$ is

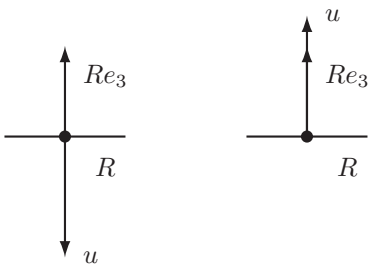


Figure 5.7: *Example 3, without a unique intersection point and ideally with* $f = ||f_r e_3 + mu||$.

needed, because $Re_3 = R_d e_3$. If, with the same initial hovering attitude, the target position is placed such that $Re_3 \perp R_d e_3$, the ideal thrust to track a straight line trajectory is $f = f_r$, which results in the quadrotor hovering at the initial position. Knowing the ideal thrust for the two scenarios described here, it is possible to formulate a thrust definition that scales between these two points. If the angle between e_3 and Re_3 is equal to the angle between e_3 and $R_d e_3$, the thrust should be $f = ||f_r e_3 + mu||$ as in (2.16), if the angle difference is $\pi/2$, the thrust should be $f = f_r$. Any other value in between is scaled with mu according to the angle difference. Since Re_3 is defined in the inertial frame $\mathcal{I}$ and $R_d e_3$ in the reference frame $\mathcal{R}$, the angle difference cannot be calculated directly. Therefore, to write both vectors in the same frame, let

$$\hat{f} = R_r^\top R e_3, \tag{5.6}$$

be the orientation of the thrust vector in the reference frame $\mathcal{R}$. From (4.8), it is known that $R_d e_3 = f_d$ and from Section 5.2.2 it is known that by rotating around the z axis, the vector f_d can be described in only two directions. Using this method allows for calculating the angular difference between Re_3 and f_d in a specific two-dimensional plane, in which straight line movements are to be tracked.
The exact formulation for defining the angle difference is as follows.
Let

$$z = \operatorname{atan2}(f_{d,2}, f_{d,1}), \tag{5.7}$$

be the angle of the desired thrust vector in the horizontal plane from b_1 in Figure 2.1. Then

$$R_{f_d} = \begin{bmatrix} \cos(z) & \sin(z) & 0 \\ -\sin(z) & \cos(z) & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{5.8}$$

defines a rotation matrix that rotates the vector f_d such that it can be described using only the first and last element of the three-dimensional vector. Multiplying the vectors $\hat{f}$ and f_d with this rotation matrix ensures both vectors are in the same frame, where thus f_d can be described by only its first and last element. Calculating the specific angles from now the b_1 direction in the new frame for both $\hat{f}$ and f_d results in

$$\lambda_1 = \operatorname{atan2}\left((R_{f_d}\hat{f})^\top e_1, (R_{f_d}\hat{f})^\top e_2\right) \tag{5.9a}$$

$$\lambda_2 = \operatorname{atan2}\left((R_{f_d} f_d)^\top e_1, (R_{f_d} f_d)^\top e_2\right) \tag{5.9b}$$

$$\lambda_e = \lambda_2 - \lambda_1. \tag{5.9c}$$

For the correct scaling factor, it is proposed to use the following function

$$g(\lambda_e) = \frac{1 + \cos(2\lambda_e)}{2}, \tag{5.10}$$

to achieve the thrust definition

$$f = ||f_r e_3 + g(\lambda_e) mu||. \tag{5.11}$$

As can be seen from Figure 5.8 this results in maximum values for situations where $\lambda_e = n\pi$, with $n \in \mathbb{Z}$, minimum values for situations where $\lambda_e = \frac{\pi}{2} + n\pi$ and weighted averages otherwise.

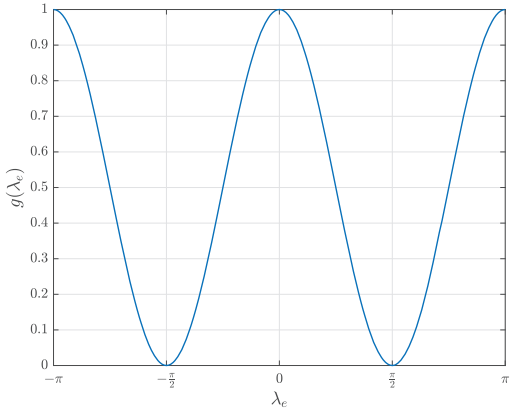


Figure 5.8: *Scaling function from* (5.10).

5.3.1 Flaws

Using the thrust definition from (5.11) does not result in great tracking performance for any scenario. For situations where a strictly positive thrust can result in an intersection with the vector $f_r e_3 + mu$, this method does not result in the behaviour described in Section 5.2, which is preferred. More flaws originate from (5.10). This function is continuous, as desired, however not linear between the extreme values which means that the error scales differently around $\lambda_e = \frac{n}{2}\pi$ compared to a discontinuous, but linearly scaling, function as

$$g(\lambda_e) = \frac{1 - ||\sin(\lambda_e)|| + ||\cos(\lambda_e)||}{2}. \tag{5.12}$$

As mentioned earlier, discontinuities were avoided for this thesis, even though improved tracking performance could be achieved with it.

5.4 Concluding remarks

In this chapter the thrust definition has been considered. A motivation has been given as to why the previous definition did not comply completely with the goal of tracking a trajectory in a straight line. The definitions from [1] and [2] were based on the assumption that the current thrust direction Re_3 is equal to the desired thrust direction $R_d e_3$ or f_d, while this is not constantly valid. Using this method resulted in behaviour that, initially, was undesirable. From Figure 1.4 it was observed that translations in the horizontal plane also caused unnecessary translations in the vertical plane, which this chapter showed to be a result of the thrust definition in (2.16).

It was therefore concluded that alternative thrust definitions should be determined, in order to prevent these unnecessary vertical translations. The first proposal was based on deriving the point of intersection between scaled versions of Re_3 and $f_r e_3 + mu$. By using this method, a perfect thrust can be obtained that exactly tracks the trajectory in a straight line. However, this method comes with multiple flaws. Flaws that can result in thrusts equal to 0, which would result in no torque and therefore no rotations. No thrust results in a free fall, and no rotation results in no possibility to change the thrust, therefore the quadrotor falls until it crashes. Another unwanted result from the method described by (5.5) is the possibility of negative thrusts. This results from the fact that intersections can also occur when scaling with negative values. It is by way of contrast not possible to generate negative thrusts. For the two unwanted scenarios therefore, a constraint for the reference thrust has been defined by [1] and [2], which combined with a saturation function limits the minimum thrust to be at least greater than 0. The final flaw resulting from defining a thrust by using matrix equalities, is caused by discontinuities. In Figure 5.7 it has been shown that no unique solution is obtainable, since the vectors are not linearly independent. When using (5.5) this means an

unsolvable equation and thus a singularity, which would result in a failing control algorithm. If the thrust vectors from Figure 5.7 would rotate clockwise or counterclockwise, these singularities would not occur, as long as the vectors stay linearly independent. Discontinuities like these can be avoided, however not in a continuous control law. It is possible that, for instance, hybrid controllers could handle these kind of discontinuities. However, since this thesis is not based on hybrid control laws but continuous control laws, discontinuities are to be avoided and another alternative had to be proposed.

The alternative proposal for the thrust definition was based on weighted averages. Reason for this was the ability to track trajectories sufficiently correct for situations where the vectors Re_3 and f_d are parallel or conversely perpendicular, when the initial attitude was hovering. For parallel vectors, the ideal thrust should be $f = ||f_r e_3 + mu||$ and for perpendicular vectors it should be $f = f_r$. It was proposed to define an angle, λ_e, that defines the error between Re_3 and f_d in order to scale the thrust between the two extreme values. The proposed scaling factor has been defined in (5.10) and the final thrust definition can be seen in (5.11). Although no discontinuities follow from this thrust definition, perfect tracking behaviour cannot be guaranteed. One of the reasons is that the scaling function does not scale in a linear fashion between the two extreme values, which can only be achieved with discontinuous scaling functions as in (5.12).

This chapter finalizes the validation and the, where necessary, adaptations of the previously defined control laws from [1] and [2]. The following chapter visualizes and compares the effectiveness of the newly defined control laws to the already existing ones by means of numerical simulations.

Chapter 6

Results

Where the previous chapters have been mainly theoretic, this chapter provides the necessary practical view. The cascaded control structure from [1] has been implemented in a simulation environment and the previously defined control laws are compared to the newly defined control laws from this thesis. By simulating the different methods, a comparison can be made which is less abstract than differences in theory. Also the behaviour of the discretized controllers can be observed. As mentioned in this thesis, singularities in the control law can occur under certain circumstances. These singularities, and the resulting behaviour during these singularities, are presented and discussed in this chapter as well.

For this to be possible however, a simulation model has to be designed first. The model is described briefly in Section 6.1 together with the specifications of the commercial drone on which this thesis is based on. Next, in Section 6.2 the actual simulation results are presented and discussed. This section consists of multiple subsections where firstly a simulation is shown in order to compare the performance of the newly designed control laws to the earlier ones shown in Chapter 1. Hereto, firstly some expectations are mentioned, after which, based on the results from this simulation, conclusions are drawn and compared to the set expectations. From these conclusions, other simulations are shown that evaluates the limits of the control laws. This chapter finishes with some concluding remarks in Section 6.3.

6.1 Simulation model

In this section the simulation model, used to achieve the coming results further on in this chapter, is provided and elaborated. The simulation program of use is Simulink and the basic layout of the simulation model is visualized in Figure 6.1. Here the block named "Control Algorithm" consists of the control laws provided in the chapters 3 through 5.

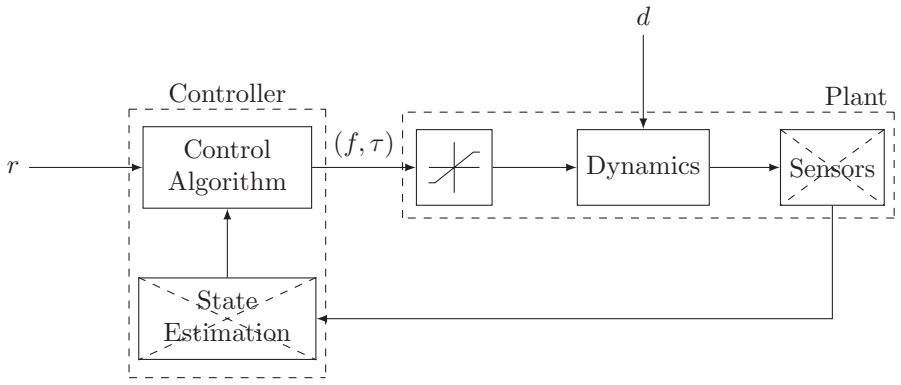


Figure 6.1: *Schematic visualization of the closed-loop model structure implemented in Simulink.*

To compare the performance of the newly described control laws to the previously defined control laws, the same basic structure of the model in Figure 6.1 is used, but implemented with the "Control Algorithm" from [1] and [2]. As explained earlier, the thrust f and torque τ are inputs to the system (2.4) coming from the control algorithm, which are saturated to comply with actuator constraints as, for instance, maximum obtainable lift [33]. Next the behaviour of the quadrotor is simulated by means of the equations provided in (2.4). Normally the translations and rotations that follow from the system dynamics are measured by sensors in an internal measurement unit (IMU). Possible sensors in an IMU can be accelerometers, magnetometers or gyroscopes. The use of sensors is inherent to the presence of measurement noise, which can cause small errors. These small errors are then fed back to the control algorithm which can then lead to undesirable inputs, if the control algorithm is not robust enough. The Input-Output linearization method from this thesis is not generally considered to be robust, as it does not account for sensor noise [34]. This, combined with the fact that this report has not paid attention to filtering measurement noise or time delay control [2], [35], is the reason that it is assumed that all states of the system are available at all times to provide a clear image of the difference between the control algorithms. Moreover, Lefeber, Greiff, and Robertsson already provide a UaGAS control law that uses only filtered signals, which is able to attenuate measurement noise. Logically, it is not possible to know all the states without using sensor data. IMUs are able to measure rotations, translations and accelerations, which provides information of three of the four total states of the system. In modern UAVs, the unavailable state is the linear velocity. Downwards facing cameras with optical flow sensors are able to measure linear velocities by integrating the motions between two frames, the downside is that optical flow sensors only work properly in certain environments. In environments such as landscapes or oceans, there are often little distinctions to be made between two frames, making it hard to calculate the flow. By using external camera devices such as OptiTrack [36] it is also possible to calculate linear velocities through integrating translations between frames. These devices are expensive solutions and require an extra internal device to connect with the UAV. State observers such as the ones designed in [12] and [2] require no additional measurements and can also accurately estimate otherwise unavailable state information. Knowing that many appropriate solutions to sensor noise and unavailable states are already available, it can be argued that implementing full state information directly from the system dynamics makes little difference to implementing alternative solutions.

Not only the control laws are of importance for correctly simulating the system, also its complementary parameters. Earlier research on the Eindhoven University of Technology provides parameters fitting to a quadrotor from Parrot called the "AR Drone 2.0". The reports from [12] and [37] are amongst those. Later research on the Eindhoven University of Technology was based on another drone from Parrot, the "Mambo Fly". These drones are considerably different from each other in not only size, the AR Drone 2.0 is three to four times the span of the Mambo Fly, but also in weight, inertia and present sensors. The table below provides more specifications of the Parrot Mambo Fly.

Table 6.1: *Overview of the sensors on the Parrot Mambo Fly.*

Sensor	
Down-facing camera	640 x 480
Front-facing camera	No
GPS	No
Barometer	Yes
Three-axis magnometer	No
Three-axis gyroscope	Yes
Three-axis accelerometer	Yes
Ultrasonic distance sensor	Yes
Optical flow sensor	Yes

It is important to visualize these differences, since this research, just as the research of [13], uses the parameters of the Parrot Mambo Fly. This because the Parrot AR 2.0 drones were no longer available. Differences between the results provided in the coming section and the results obtained by

other research is therefore explained because of the differences in drone use.

6.2 Simulation results

In this section multiple simulations are performed and their corresponding results are shown and discussed. These simulations are done in order to compare the performance of the newly designed control laws to the already existing ones from [2] in closed-loop. Both the expected strengths and weaknesses are discussed beforehand, after which the simulations either confirm or oppose these expectations. As there have not been made adjustments to the position control, no specific attention has been given to that matter. It can be assumed that the position control law from (3.11) is implemented in every control algorithm.

For the following simulations, the initial conditions from (6.1) are used.

$$\rho_0 = \begin{bmatrix} 4 & 4 & -1.5 \end{bmatrix}^\top \qquad \nu_0 = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^\top \qquad R_0 = I \qquad \omega_0 = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^\top. \tag{6.1}$$

The coming simulations all describe a time dependent reference position trajectory, which is defined as

$$\rho_r(t) = \begin{bmatrix} \cos(\omega_v t) & \sin(\omega_v t) & -1.5 \end{bmatrix}^\top, \tag{6.2}$$

with $\omega_v = 0.2\pi$. Time dependent trajectories require a specific method to determine the other correct, time dependent, references. It is possible to obtain the references of the complete system, as in (2.4), from only a few flat outputs. This method does not require integration of references, but is dependent on the smoothness of the desired trajectory. For the trajectory to be sufficiently smooth means it should be differentiable at least as many times needed to define all the system states from these derivatives [16]. The exact method to rewrite flat output trajectories to full state references is provided in detail in Appendix B. To also track a specific target reference position for the yaw angle, the following position is set

$$\rho_t = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^\top, \tag{6.3}$$

which is the center of the rotations in the horizontal plane described in (6.2). Note that it is of no importance how the third entry for ρ_t is chosen, as it is not necessary from the definition in (4.13). Additionally, the control parameters used for all simulations can be seen in Table 6.1.

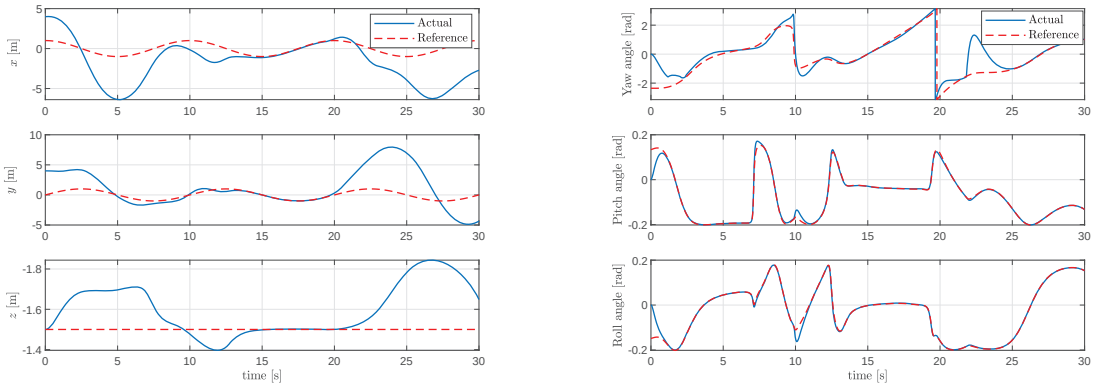
Table 6.1: *Control parameters used in simulations.*

Parameter	Description	Value
(k_p, k_d)	Translational control gains	(4,4)
γ	Saturation bound	2
($k_{p,r,1}$, $k_{d,r,1}$)	Yaw control gains	(4,4)
($k_{p,r,2}$, $k_{d,r,2}$)	Pitch control gains	(25, 10)
($k_{p,r,3}$ $k_{d,r,3}$)	Roll control gains	(25,10)
ϵ	Scalar bound preventing Gimbal Lock	0.1

6.2.1 Control laws with target heading direction

By comparing the already existing control laws to the newly designed ones in this report, it is possible to make distinctions in behaviour. Earlier in Chapter 1, it was shown that the transient behaviour denies translations in a straight path towards the target and Chapter 5 showed that the thrust definition caused unnecessary altitude errors, which were also visible in the figures from Chapter 1. It is expected that these results show increased tracking performance, when compared to the results obtained in Figure 1.3, in that the drone follows a straight path towards the reference position. This follows from the Input-Output linearization and the desired tracking dynamics. It is also expected that there is little to no overshoot during transient behaviour, since the attitude control is not designed to be agile and therefore capable of difficult maneuvers. Instead, it is designed to track references without having unnecessary translations. This means that the new attitude control law

converges slower to the reference orientations and therefore also slower to the reference position. For this reason, little overshoot is expected. As for the altitude changes, they are expected to decrease, albeit slightly. This because the newly designed thrust definition is also a slight improvement when compared to the thrust definition from [1]. Lastly, the yaw angle is expected to rotate exactly as desired, making it face the reference orientation continuously. From Figure 6.4,it is clear that the



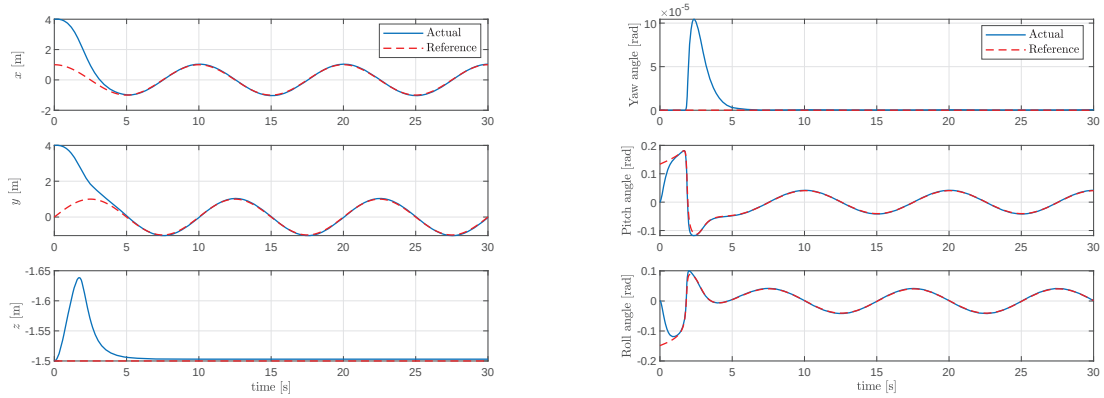
(a) *Position tracking response.*

(b) *Attitude tracking response.*

Figure 6.2: *Tracking performance using the newly designed control laws.*

control laws do not perform as expected. Even though it seems like the most important rotation angles, ϕ and θ, show decent tracking performance, it is not visible in the position tracking. The only points where the tracking performance is good, is from $t \approx 15$ until $t \approx 20$. This is also the range where the tracking of the yaw angle is as desired. It is around the points where the yaw angle crosses $\psi = \pi$ to $\psi = -\pi$, where the system seems to lose control. The fact that the system loses control from yaw rotations specifically is unexpected, as these rotations should not have an influence on the other rotation angles, since those translations are decoupled using the Input-Output linearization. It is clear that around $t = 10$, not only the yaw tracking performance is bad, but also the roll and pitch tracking performance, even though they should not be influenced by the yaw rotations. It is possible that minor errors in the decoupling of the system in the simulation are the cause of this, as it should not happen. Contradictory however, when the yaw tracking performance decreases around $t = 22$, tracking performance in roll and pitch do not decrease. This might indicate that there are other mistakes at cause. Focusing on the point around $t = 19$, the tracking performance is good in both attitude and position and the yaw reference nears the crossing from π to $-\pi$. It can be observed that suddenly the roll and pitch references, and therefore also the actual roll and pitch angles, divert from the angles they had before. This seems unnecessary, as the tracking performance was good until this point. From this observation, it is expected that possibly there have been made mistakes in either the implementation of the reference attitude angles or the definition of these reference angles from the pseudo code in Appendix A. Then the yaw tracking performance itself. It can be noticed that one singularity resulting from the yaw reference has not been discussed yet. When the reference yaw angle crosses the point where $\psi = \pi$ to $\psi = -\pi$, it can be observed that the actual yaw angle is somewhat ahead of the reference angle and therefore crosses this point earlier than the reference. As a result, the yaw error suddenly increases to approximately π. This in turn leads to a response of the yaw angle to return to $\psi = \pi$. During this transition however, the reference yaw angle crosses from $\psi_r = \pi$ to $\psi_r = -\pi$, which again results in the actual yaw angle to quickly return to that specific reference. This behaviour is undesirable and is to be improved.

In order to prove that most of the errors result from the singularity in the yaw reference and that the other control laws do result in the, expected, desirable behaviour, a new simulation has been performed. Now, instead of the target reference position from (6.3), the yaw reference angle is set to 0 at all times, and therefore also the derivatives of this yaw angle. Except for this, no other references have been changed. From Figure 6.3 is can be observed that the absence of a yaw reference improves the tracking performance significantly. The behaviour in this figure resembles the expectations men-



(a) *Position tracking response.*

(b) *Attitude tracking response.*

Figure 6.3: *Tracking performance using the newly designed control laws and no yaw references.*

tioned earlier, in that there is little to no overshoot in the tracking of the position references. The corresponding attitude tracking also performs well, with a minor flaw in the yaw angle around $t = 2$. Furthermore, it can be noticed that after about six seconds, the system is converged. However, it can also be seen that after these six seconds there remain some minor errors in both x and y rotations and translations, but also in the altitude. This is the result from the Input-Output linearization method being too slow for the references. It is possible that these errors decrease or even disappear if the control gains are tuned better.

It can also be observed that the altitude translations behave differently when compared to the altitude changes in Figure 1.3. Instead of first increasing and then decreasing, the altitude now only increases, after which it converges to the reference position in a critically damped manner. Unfortunately, the peak error is now about ten centimeters bigger than in Figure 1.3. However, as the newly defined thrust definition from (5.11) scales according to the error between the achievable thrust vector and the desired thrust vector and the thrust definition from (2.16) does not, it is still expected that the newly defined thrust definition performs better than the one from [1].

To conclude if the new control laws result in improved transient translational behaviour, Figure 6.4 shows the x and y translations. When compared to the results obtained in Figure 1.3a, it can be observed that the drone translates in a near straight line towards the time dependent reference position. Now the advantage of the "slow" Input-Output linearization technique can be seen. Since it rotates, and thus translates, slower, it is also less influenced by the changing references in transient behaviour. Now it can also be seen more clearly that there remain some steady state errors in the position. Figure 6.5 illustrates that this steady state error originates from the minor errors that remain in the orientation. For more information on the error definition used in this figure, refer to Chapter 2.

6.2.2 Singularities

Besides the singularity coming from the yaw reference angle around π and $-\pi$, Chapter 4 also discussed another singularity. This one should arise when the position of the drone in the horizontal plane is equal to the target reference position for the yaw definition. This singularity is expected to disrupt the complete system, leading it into instability, which is highly unwanted. The following simulation presents the behaviour of the drone, when it would translate through the target reference position for the yaw reference. For this simulation the following, time independent, references have been set, together with the initial conditions from (6.1)

$$\rho_r = \begin{bmatrix} 8 & 4 & -1.5 \end{bmatrix}^\top \qquad \nu_r = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^\top \qquad R_r = I \qquad \omega_r = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^\top. \tag{6.4}$$

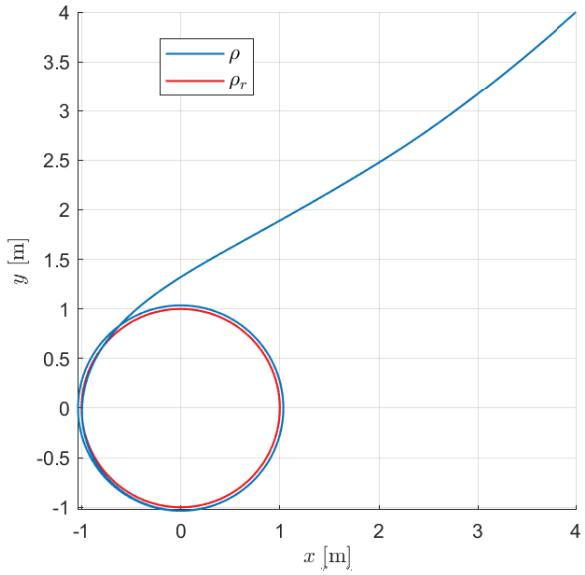


Figure 6.4: *Quadrotor position and reference position, using new methods.*

The target reference position for the yaw has been set to be

$$\rho_t = \begin{bmatrix} 6 & 4 & 0 \end{bmatrix}^\top . \tag{6.5}$$

From Figure 6.6 it can be observed that, because of the Input-Output linearization, only rotations around the x-axis occur in order to translate in the y-direction. This happens as desired until $t \approx 2$, since at this point ρ is almost identical to ρ_t. It is not exactly identical, because otherwise the simulation would have failed since there would be a singular value for ψ_r. However, even though no singular values enter the system, the system does become unstable. This is the result of the undesirable behaviour described using Figure 4.3, where in a short amount of time the yaw error changes from zero to π, causing extreme, and possibly unachievable, control efforts. Notice how it is possibly unachievable, since no time has been dedicated to rewrite the control outputs to actual thrusts per rotor as in (2.7).

From these results, it can be concluded that careful thought has to be dedicated to choosing the target reference point. If the reference position and the target reference position for the yaw could become identical, these singularities occur. As mentioned in Section 4.3, by using collision avoidance methods in combination with the yaw definition, it would never be possible for the position to come near the target reference position and therefore the unstable behaviour would be avoided.

6.3 Concluding remarks

In this chapter, the newly proposed control laws have been simulated and compared to control laws that were already available, with the aim to visualize the differences in behaviour and to validate which control laws would be most suitable for the goal of this research. Hereto, a numerical model has been developed in Simulink. This model is designed such that all the states are directly available in the Control Algorithm, which has been illustrated in Figure 6.1. The reason for this came from the Input-Output linearization method designed in this thesis being not robust to measurement noise [34] due to the absence of methods that use filtered signals or time delay control [2][35]. Another method to increase robustness of the feedback linearization can be to implement direct adaptive feedback linearization techniques, as in [38], in which uncertain system parameters and external disturbances can be corrected for. Multiple simulations have been performed, each with the intent to highlight the strengths and weaknesses of the newly designed control laws from this thesis. The

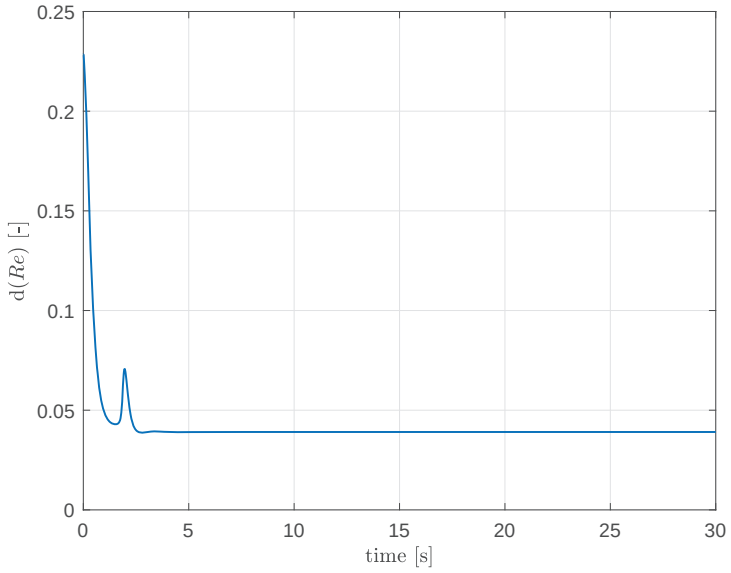


Figure 6.5: *Distance to attitude control error equilibrium.*

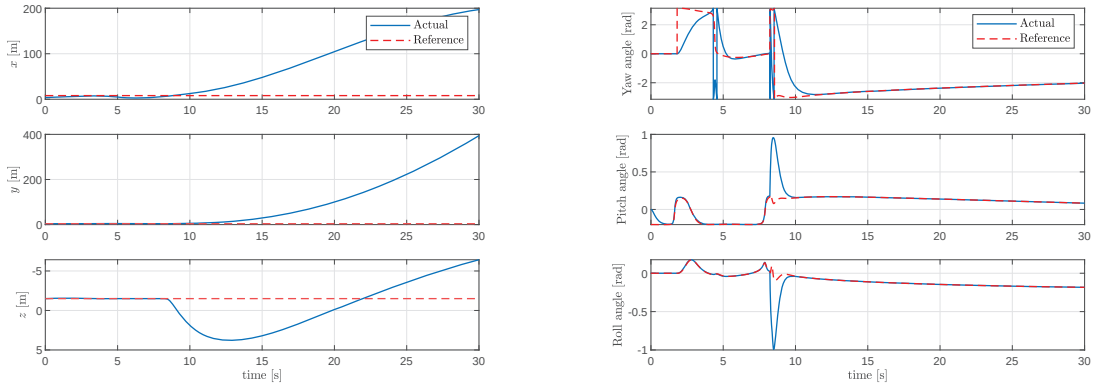
position control laws have been ignored, as the already available position control laws already sufficed.

First of all, simulations were presented where the complete set of tracking control laws were implemented, including the new yaw definition. The results in Figure 6.2 showed undesirable tracking performance when the target reference position was set to be inside the reference trajectory. This was because the yaw definition contained a singularity that had not been accounted for, namely the transition point from $\psi_r = \pi$ to $\psi_r = -\pi$. If the actual yaw angle was slightly ahead of the reference and transitioned earlier to $\psi = -\pi$ than the reference, it tried to correct for the immediate yaw error by rotating back to $\psi = \pi$. However, during this transition, the reference yaw transitions to $\psi_r = -\pi$, which leads to the yaw angle rotating back again. These short rotations seemed to have an impact on the roll and pitch angles as well, which could be because of a minor error in the implementation of the Input-Output linearization. However, it could also be the result of a mistake in the implementation of the reference attitude angles or in the definition of these reference angles from the pseudo code in Appendix A.

Next, by presenting another simulation where the yaw references and its corresponding derivatives were set to zero, it was shown that the control laws can perform as desired. Figure 6.3 illustrated that the rotations and positions converge to the desired references, albeit with a steady state error resulting from the relatively slow Input-Output linearization method. It has also been shown that the altitude error has increased compared to the results obtained in Chapter 1. However, since the new thrust definition compensates for the angle difference in desired and actual thrust vector, whereas the earlier one from [1] does not, it can be concluded that the new thrust definition still performed better than the earlier one. Figure 6.4 also showed that the new control laws perform better during transient behaviour, in that the drone translates in a near straight line towards the reference position.

Lastly, simulations have been performed where the expected singularities from Section 4.3 were highlighted. It has been shown that the system can become unstable when the actual position and the target reference position are identical. If this situation occurs during a measurement of the simulation, the system becomes unstable immediately. Figure 6.6 showed that if this situation occurs between two measurement points, the large, and possibly unachievable, control efforts that are necessary to compensate for these errors cause the system to become unstable too.

These results and the results from the previous chapters are discussed in more detail in the following



(a) *Position tracking response.*

(b) *Attitude tracking response.*

Figure 6.6: *Tracking performance using the newly designed control laws and a yaw reference that can become singular.*

chapter, where conclusions are drawn and recommendations for future work are suggested.

Chapter 7

Conclusions and Recommendations

This thesis aims at improving the tracking control problem for a quadrotor. Earlier designed nonlinear control laws have been validated and alternatives have been proposed for the control laws which did not result in the desired behaviour for this thesis, namely to improve transient translations and rotations by flying in a straight path towards a certain target. This chapter provides the necessary conclusions that can be drawn from this thesis. Additionally, some recommendations for possible future work are presented.

7.1 Conclusions

Firstly, the main results of this thesis are revisited and corresponding conclusions are drawn in the same order as they have been addressed: from position control, to attitude control, to combined control. Hereto, the theoretical research on the modeling and control of the quadrotor are stated at first and subsequently the simulations.

Modelling and Control

The position control law designed by [1] has been validated. In order to do so, the desired position tracking dynamics have been described and the corresponding controllers to achieve those desired position tracking dynamics have been designed, firstly for a reference without attitude dynamics and lastly for a time dependent reference orientation. The final controller was identical to the one described in [2], confirming that no alternatives had to be designed for these dynamics. Earlier observed, undesired, behaviour therefore did not originate from this controller.

Initial experiments showed undesirable behaviour following from the attitude dynamics. Specific orientations that could be achieved by rotating along a single axis towards the reference orientation would rotate along multiple axis, leading to unnecessary translations. For this reason, an alternative for the attitude tracking control has been proposed based on Input-Output linearization techniques. By using Input-Output linearization the desired rotations were achieved. These desired rotations were derived from the matrix R_d and a rotation matrix based on rotations around roll and pitch only, leaving the yaw free to be specified individually. Input-Output linearization is not a common method to define a control law, mainly because of the sensitivity to sensor noise. Little sensor noise can result in unstable behaviour. Especially in Input-Output linearization, because faulty state measurements result in incorrect decoupling terms and therefore faulty tracking control laws [34]. The new definition for the yaw angle enabled tracking of a constant point in the horizontal plane, which was one of the goals of this thesis. It has been proven that this definition could be used as a flat output, since the trajectory is sufficiently smooth. Singularities arose using this definition, but in Section 7.2 a solution is provided to counteract these singularities. Singularities also arose in the tracking dynamics for the attitude control, in the form of Gimbal Lock. This was a result of defining these tracking controls in Euler angles initially, after which they were rewritten to $SO(3)$. These singularities can be overcome by using (4.12) from [31].

By means of schematic representations of the drone with the directions and magnitudes of the involved forces and accelerations, it is shown that the thrust definition from [1] did not result in the desired tracking behaviour for this thesis. Using their definition resulted in undesired altitude changes if the current thrust vector, Re_3, did not align with the desired thrust vector, f_d. Two alternatives were proposed. The former was based on calculating the exact geometrically ideal solution, but, due to singularities, did not meet the desire of being a continuous control law. The latter used scaled values for the desired acceleration direction, mu, that follow from the angular differences between Re_3 and f_d in a two-dimensional plane. This method is continuously defined and was a step forward from the previously defined thrust definition.

Simulations

The control laws proposed in this thesis have been implemented in a numerical model of a quadrotor, this numerical model included parameters corresponding to a Parrot Mambo Fly. The already existing control laws with corresponding gains have been equipped with the same parameters to study the differences. Initial simulations where the complete set of newly designed control laws were implemented, showed undesirable tracking results. It was observed that this was either a result of a mistake in the implementation of the Input-Output linearization or a mistake in the implementation, or definition, of the reference attitude angles resulting from the pseudo code in Appendix A. A combination of both is not excluded either. By leaving out the yaw reference definition from (4.13) and its derivatives, the undesirable behaviour did not arise. This meant conclusions could be drawn from the other control laws. These simulations showed the expected, desired, behaviour. It was shown that the rotations and positions converged to their respective references, although minor steady state errors remained present. These minor errors were expected to be a result of the relatively "slow" Input-Output linearization method. Even though the altitude changes seemed to increase when compared to the results using the control laws from earlier work, the new thrust definition was still concluded to be an improvement. This because the new thrust definition compensates for the angle difference in desired and actual thrust vector, whereas the earlier one from [1] does not.

The goal of this research was to improve transient behaviour, which has been achieved to a certain degree for the attitude control. Although the new combined control law improved on the definition from [1], it has not fully achieved the desire to translate in a straight line. This does however raise the question: "*Is it even possible to design a continuous control law that enables straight path tracking dynamics?*" Future research may answer this question.

7.2 Recommendations

Finally, the following section provides recommendations for future work. These recommendations are presented in an identical order as for the conclusions. That is, firstly the modelling is discussed and subsequently the simulations.

Modelling

As stated earlier, the attitude dynamics now have not been presented in a realistic manner. Mainly because of the absence of sensors and therefore state estimators. These additions decrease the tracking performance of the control law, since sensor noise and estimation errors enter the control algorithm. Lefeber, Greiff, and Robertsson (2020) presented a tracking controller for quadrotor UAVs which uses partial state information and filters measurements to attenuate noise, while being closed-loop uniformly almost globally asymptotically stable and locally exponentially stable. Implementing these methods in the current control algorithm could limit the decrease in real-life tracking control performance. Another method to improve the accuracy of the numerical model of the quadrotor is to account for drone specific damping forces, such as blade flapping and drag. Brekelmans (2019) provided extensive information on this matter.

As for improving the control method, multiple suggestions are provided. First of all, the Input-Output linearization method can be made more robust by implementing direct adaptive feedback linearization techniques as in [38]. Using this technique, the controllers can correct for errors in system parameters or external disturbances. Other options can be to use filtered signals or time delay control [2], [35]. Subsequently, the yaw angle definition can be improved. The results showed that undesirable behaviour arose when the yaw angle is approximately π. This behaviour arises because of singularities resulting from using Euler angles. It is proposed therefore, to examine possibilities to avoid these singularities. Greiff (2017) implemented such a method to avoid gimbal lock, implementing a similar method in the yaw definition can result in proper behaviour. Future work might focus on designing a similar, yet different, method or focus on rewriting this definition for $SO(3)$ completely. Another singularity that was discussed shortly in Chapter 4.3, can result from the situation where the target heading position and the current position are identical. For this position, no heading direction exists. Therefore, it is preferred to avoid the situation where the target heading position and the drone position are identical. A fitting solution can be to implement a collision avoidance method as in [32] (2020), where not another agent or object is to be avoided, but the target heading direction. This would guarantee that the target heading position and the drone position are never identical, because it translates around this point, and therefore no singularities can occur. One possible method to extend this research would be to implement additional control laws that enable the usage of multiple drones. For monitoring purposes, more drones would mean more vision on the specific target area, which could be of interest. In the work of [37] and [39], different control laws that achieve tracking dynamics for multiple drones is provided.

At last, it would be of the most importance to prove the stability of this tracking controller for both time independent as time dependent reference trajectories.

Simulation

As mentioned in the conclusions already, the absence of sensors and therefore sensor noise fails to provide a realistic view to the current control law and as a result of that, also the simulation. The numerical model should therefore be implemented with sensor noise similar to the noise that arises when using the Parrot Mambo Fly.

Also, it has to be noted that physical limitations of each rotor are not considered in this thesis. The work of de Jonge (2020) provides a matrix, (2.7), to convert the thrust and torque inputs to individual thrust forces which subsequently can be rewritten to rotor speeds and therefore required pulse-width modulations (PWMs) for the actual drone or the numeric representation of the drone. If the inputs result in rotor speeds or PWMs that are physically unobtainable, the actual output can not result in desired behaviour. The simulation used in this thesis did not account for such limitations, possible future research could.

At last, it is clear that the most realistic simulation is an actual experiment. Unfortunately, due to lack of time, extensive experiments have not been conducted. Future work could therefore implement the tracking control law on an actual Parrot Mambo Fly, using the Simulink support package and additional support provided by [13] to draw conclusions from the results that follow from those experiments.

Bibliography

[1] E. Lefeber, S. van den Eijnden, and H. Nijmeijer, "Almost global tracking control of a quadrotor UAV on SE(3)," in *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, 2017, pp. 1175–1180. DOI: 10.1109/CDC.2017.8263815.

[2] E. Lefeber, M. Greiff, and A. Robertsson, "Filtered Output Feedback Tracking Control of a Quadrotor UAV," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 5764–5770, 2020, 21th IFAC World Congress, ISSN: 2405-8963. DOI: https://doi.org/10.1016/j.ifacol.2020.12.1609. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405896320322059.

[3] G. Udeanu, A. Dobrescu, and M. Oltean, "Unmanned Aerial Vehicle in Military Operations," *Scientific Research And Education In The Air Force*, vol. 18, pp. 199–206, Jun. 2016. DOI: 10.19062/2247-3173.2016.18.1.26.

[4] R. Andaru, J. Rau, D. Syahbana, A. Prayoga, and H. Purnamasari, "The use of UAV remote sensing for observing lava dome emplacement and areas of potential lahar hazards: An example from the 2017-2019 eruption crisis at Mount Agung in Bali," *Journal of Volcanology and Geothermal Research*, vol. 415, p. 107 255, Apr. 2021. DOI: 10.1016/j.jvolgeores.2021.107255.

[5] S. Inc., *Drones - worldwide*, 2022. [Online]. Available: https://www.statista.com/outlook/cmo/consumer-electronics/drones/worldwide (visited on 04/19/2022).

[6] Amazon, *Amazon Air launches new hub in Southern California*, 2021. [Online]. Available: https://www.aboutamazon.com/news/transportation/amazon-air-launches-new-hub-in-southern-california (visited on 06/21/2021).

[7] E. Lamptey and D. Serwaa, "The Use of Zipline Drones Technology for COVID-19 Samples Transportation in Ghana," *HighTech and Innovation Journal*, vol. 1, no. 2, pp. 67–71, Jun. 2020. DOI: 10.28991/hij-2020-01-02-03. [Online]. Available: www.HighTechJournal.org.

[8] P. Radoglou Grammatikis, P. Sarigiannidis, T. Lagkas, and I. Moscholios, "A Compilation of UAV Applications for Precision Agriculture," *Computer Networks*, vol. 172, p. 107 148, Feb. 2020. DOI: 10.1016/j.comnet.2020.107148.

[9] N. Mohamed, J. Al-Jaroodi, I. Jawhar, A. Idries, and F. Mohammed, "Unmanned aerial vehicles applications in future smart cities," *Technological Forecasting and Social Change*, vol. 153, p. 119 293, 2020, ISSN: 0040-1625. DOI: https://doi.org/10.1016/j.techfore.2018.05.004. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0040162517314968.

[10] E. Seifert, S. Seifert, H. Vogt, *et al.*, "Influence of Drone Altitude, Image Overlap, and Optical Sensor Resolution on Multi-View Reconstruction of Forest Images," *Remote Sensing*, vol. 11, p. 1252, May 2019. DOI: 10.3390/rs11101252.

[11] A. Bhardwaj, L. Sam, Akanksha, F. J. Martín-Torres, and R. Kumar, "UAVs as remote sensing platform in glaciology: Present applications and future prospects," *Remote Sensing of Environment*, vol. 175, pp. 196–204, 2016, ISSN: 0034-4257. DOI: https://doi.org/10.1016/j.rse.2015.12.029. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0034425715302509.

[12] S. van den Eijnden, "Cascade based tracking control of quadrotors," DC 2017.012, MSc Thesis, Eindhoven University of Technology, Dynamics and Control Group, Department of Mechanical Engineering, Eindhoven, The Netherlands, 2017.

[13] G. H. Brekelmans, "Extended Quadrotor Dynamics: from Simulations to Experiments," DC 2019.090, MSc Thesis, Eindhoven University of Technology, Dynamics and Control Group, Department of Mechanical Engineering, Eindhoven, The Netherlands, 2019.

[14] D. Eberly, *Rotation Representations and Performance Issues*, Technical report, Geometric Tools, Redmond WA 98052, Jan. 2002. [Online]. Available: https://www.geometrictools.com/Documentation/RotationIssues.pdf.

[15] N. Jeurgens, "Identification and control implementation of an AR.Drone 2.0," DC 2017.013, MSc Thesis, Eindhoven University of Technology, Dynamics and Control Group, Department of Mechanical Engineering, Eindhoven, The Netherlands, 2019.

[16] M. Fliess, J. Lévine, P. Martin, and P. Rouchon, "On differentially flat nonlinear systems," *IFAC Proceedings Volumes*, vol. 25, no. 13, pp. 159–163, 1992, 2nd IFAC Symposium on Nonlinear Control Systems Design 1992, Bordeaux, France, 24-26 June, ISSN: 1474-6670. DOI: https://doi.org/10.1016/S1474-6670(17)52275-2. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1474667017522752.

[17] B. K. P. Horn, "Closed-form solution of absolute orientation using unit quaternions," *J. Opt. Soc. Am. A*, vol. 4, no. 4, pp. 629–642, Apr. 1987. DOI: 10.1364/JOSAA.4.000629. [Online]. Available: http://opg.optica.org/josaa/abstract.cfm?URI=josaa-4-4-629.

[18] K. W. Spring, "Euler parameters and the use of quaternion algebra in the manipulation of finite rotations: A review," *Mechanism and Machine Theory*, vol. 21, no. 5, pp. 365–373, 1986, ISSN: 0094-114X. DOI: https://doi.org/10.1016/0094-114X(86)90084-4. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0094114X86900844.

[19] N. van de Wouw, *Multibody and Nonlinear Dynamics, course slides*, Eindhoven University of Technology, 2021.

[20] M. Kissai, B. Monsuez, X. Mouton, D. Martinez, and A. Tapus, "Adaptive robust vehicle motion control for future over-actuated vehicles," *Machines*, vol. 7, p. 26, Apr. 2019. DOI: 10.3390/machines7020026.

[21] N. Hall, *Aircraft rotations*, 2022. [Online]. Available: https://www.grc.nasa.gov/WWW/K-12/airplane/rotations.html (visited on 04/26/2022).

[22] E. V. Lewis, *Principles of Naval Architecture*. 1989, vol. 3, p. 41, ISBN: 0939773023.

[23] V. Lepetit and P. Fua, "Monocular model-based 3d tracking of rigid objects: A survey," *FoundationsÂ and TrendsÂ® in Computer Graphics and Vision*, vol. 1, no. 1, pp. 1–89, 2005, ISSN: 1572-2740. DOI: 10.1561/0600000001. [Online]. Available: http://dx.doi.org/10.1561/0600000001.

[24] H. K. Khalil, *Nonlinear systems; 3rd ed.* Upper Saddle River, NJ: Prentice-Hall, 2002, The book can be consulted by contacting: PH-AID: Wallet, Lionel. [Online]. Available: https://cds.cern.ch/record/1173048.

[25] H. Sira-Ramirez and S. Agrawal, "Differentially flat systems," vol. 17, Jan. 2004.

[26] M. Heemels, M. Chong, T. Meijer, *et al.*, *System Theory for Control, course slides*, Eindhoven University of Technology, 2021.

[27] A. Benallegue, A. Mokhtari, and L. Fridman, "Feedback linearization and high order sliding mode observer for a quadrotor uav," in *International Workshop on Variable Structure Systems, 2006. VSS'06.*, 2006, pp. 365–372. DOI: 10.1109/VSS.2006.1644545.

[28] A. Das, F. Lewis, and K. Subbarao, "Backstepping approach for controlling a quadrotor using lagrange form dynamics," *Journal of Intelligent and Robotic Systems*, vol. 56, pp. 127–151, Sep. 2009. DOI: 10.1007/s10846-009-9331-0.

[29] K. Alexis, G. Nikolakopoulos, and A. Tzes, "Model predictive quadrotor control: Attitude, altitude and position experimental studies," *Control Theory & Applications, IET*, vol. 6, pp. 1812–1827, Aug. 2012. DOI: 10.1049/iet-cta.2011.0348.

[30] H. Nijmeijer, E. Lefeber, and S. van den Eijnden, *Nonlinear Control, course slides*, Eindhoven University of Technology, 2020.

[31] M. Greiff, *Modelling and Control of the Crazyflie Quadrotor for Aggressive and Autonomous Flight by Optical Flow Driven State Estimation*, eng, Student Paper, 2017.

[32] W. de Jonge, "Social behavior in a network of UAVs with collision avoidance," DC 2020.093, MSc Thesis, Eindhoven University of Technology, Dynamics and Control Group, Department of Mechanical Engineering, Eindhoven, The Netherlands, 2020.

[33] F. Mazenc, R. Mahony, and R. Lozano, "Forwarding control of scale model autonomous helicopter: A lyapunov control design," in *42nd IEEE International Conference on Decision and Control (IEEE Cat. No.03CH37475)*, vol. 4, 2003, 3960–3965 vol.4. DOI: 10.1109/CDC.2003.1271769.

[34] J. P. Hespanha, *Linear systems theory*. 2009, Cited by: 455. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-77957760998%5C&partnerID=40%5C&md5=131c9bed170e3d5aae8569a3820dc853.

[35] K. Youcef-Toumi and S.-T. Wu, "Input/output linearization using time delay control," in *1991 American Control Conference*, 1991, pp. 2601–2606. DOI: 10.23919/ACC.1991.4791872.

[36] Optitrack, *OptiTrack for Robotics*, https://optitrack.com/applications/robotics/. (visited on 05/12/2022).

[37] M. van de Westerlo, "Formation tracking with multiple mutually coupled quadrotor UAVs on SE(3)," DC 2019.002, MSc Thesis, Eindhoven University of Technology, Dynamics and Control Group, Department of Mechanical Engineering, Eindhoven, The Netherlands, 2019.

[38] P. Mukherjee and S. L. Waslander, "Direct adaptive feedback linearization for quadrotor control," Cited by: 29, 2012. DOI: 10.2514/6.2012-4917. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85088757107%5C&doi=10.2514%5C%2f6.2012-4917%5C&partnerID=40%5C&md5=45af2c58ff36bed44cef401c43c79fa9.

[39] M. Zhang, Y. Shen, Q. Wang, and Y. Wang, "Dynamic artificial potential field based multi-robot formation control," *2010 IEEE International Instrumentation and Measurement Technology Conference, I2MTC 2010 - Proceedings*, pp. 1530–1534, Jan. 2010. DOI: 10.1109/IMTC.2010.5488238.

Appendix A

Input-Output terms

In this appendix, more insight is given to the Input-Output linearization method from Section 4.2.1. Hereto, the Input-Output linearization terms are presented first in Euler angles. Next, their conversions from Euler angles to $SO(3)$ are shown, and finally a pseudo code to retrieve Euler angles from a rotation matrix.

A.1 Relative degrees of outputs

Validating if the relative degrees of y_i in (4.3), $\sum_{i=1}^{3} k_i \leq n$, means calculating Lie derivatives until $L_G L_f^{k-1} h(x) \neq 0$ for each output. Note that this research uses a $SO(3)$ parametrization, but for the sake of clarity the Euler angle notation is presented, whereas in the following section a method to rewrite the definitions to $SO(3)$ is described.

First of all, from [31], let

$$\omega = R(\psi)\begin{bmatrix}\dot{\psi}\\0\\0\end{bmatrix} + R(\psi)R(\theta)\begin{bmatrix}0\\\dot{\theta}\\0\end{bmatrix} + R(\psi)R(\theta)R(\phi)\begin{bmatrix}0\\0\\\dot{\phi}\end{bmatrix} = \begin{bmatrix}-s\theta & 0 & 1\\ c\theta s\phi & c\phi & 0\\ c\theta c\phi & -s\phi & 0\end{bmatrix}\begin{bmatrix}\dot{\psi}\\\dot{\theta}\\\dot{\phi}\end{bmatrix}, \tag{A.1}$$

which can be rewritten to

$$\begin{bmatrix}\dot{\psi}\\\dot{\theta}\\\dot{\phi}\end{bmatrix} = L_f h(x) + L_G h(x) = \begin{bmatrix}0 & \frac{s\phi}{c\theta} & \frac{c\phi}{c\theta}\\ 0 & c\phi & -s\phi\\ 1 & s\phi t\theta & c\phi t\theta\end{bmatrix}\omega. \tag{A.2}$$

As $L_G h(x)u = 0$, another Lie derivative has to be calculated. Now, (A.2) is used directly to find

$$\begin{aligned}\begin{bmatrix}\ddot{\psi}\\\ddot{\theta}\\\ddot{\phi}\end{bmatrix} = L_f^2 h(x) + L_G L_f h(x) &= \frac{d}{dt}\begin{bmatrix}0 & \frac{s\phi}{c\theta} & \frac{c\phi}{c\theta}\\ 0 & c\phi & -s\phi\\ 1 & s\phi t\theta & c\phi t\theta\end{bmatrix}\omega + \begin{bmatrix}0 & \frac{s\phi}{c\theta} & \frac{c\phi}{c\theta}\\ 0 & c\phi & -s\phi\\ 1 & s\phi t\theta & c\phi t\theta\end{bmatrix}\dot{\omega}\\ &= M(\dot{\psi},\dot{\theta},\dot{\phi},\omega)\omega + \begin{bmatrix}0 & \frac{s\phi}{c\theta} & \frac{c\phi}{c\theta}\\ 0 & c\phi & -s\phi\\ 1 & s\phi t\theta & c\phi t\theta\end{bmatrix} J^{-1}\left[S(J\omega)\omega + \tau\right],\end{aligned} \tag{A.3}$$

where $M(\dot{\psi},\dot{\theta},\dot{\phi},\omega)$ results from the derivative of the matrix in (A.2)

$$M(\psi,\theta,\phi,\dot{\psi},\dot{\theta},\dot{\phi}) = \frac{d}{dt}\begin{bmatrix}0 & \frac{s\phi}{c\theta} & \frac{c\phi}{c\theta}\\ 0 & c\phi & -s\phi\\ 1 & s\phi t\theta & c\phi t\theta\end{bmatrix} \tag{A.4}$$

$$= \begin{bmatrix}0 & \frac{\dot{\phi}c\phi c\theta + \dot{\theta}s\phi s\theta}{c\theta^2} & -\frac{\dot{\phi}c\theta s\phi - \dot{\theta}c\phi s\theta}{c\theta^2}\\ 0 & -\dot{\phi}s\phi & -\dot{\phi}c\phi\\ 0 & \frac{\dot{\theta}s\phi + \dot{\phi}c\phi c\theta s\theta}{c\theta^2} & \frac{\dot{\theta}c\phi - \dot{\phi}c\theta s\phi s\theta}{c\theta^2}\end{bmatrix}. \tag{A.5}$$

Note that ω can be rewritten to $\psi, \theta, \phi, \dot{\psi}, \dot{\theta}$ and $\dot{\phi}$ using (A.1).

From the presence of τ, it can be concluded that $L_G L_f h(x) \neq 0$ and therefore that $k_i = 2$. This conclusion follows from the fact that τ is an input from (2.4d). The total sum of the relative degrees is found to be $\sum_{i=1}^{3} k_i = 6$. The system is of the 6^{th} order and therefore has 6 states. It can thus be concluded that $\sum_{i=1}^{3} k_i = n$ which implies the absence of zero dynamics. For more information on zero dynamics, see Chapter 2.

A.2 Conversion between Euler angles and $SO(3)$

To convert the decoupling terms to the desired parametrization a few steps are necessary. First of which is to decide for which situation the conversion is needed. It can be either for determining the current states of the system, or for determining the required reference state. Both situations follow the same method, but with different variables. After deciding for which cause the conversion is needed, either the rotation matrix from (2.1) is used or the rotation matrix from (4.10), where the former is needed for defining the current attitude state and the latter for determining the required reference attitude state. Using either (2.1) or (4.10), the derivative of R can be found through differentiation. From this expression, it is possible to select a specific entry of $\dot{R}$, where a single Euler derivative term can be found. Rewriting the resulting expression from $\dot{R}$ as a function of Euler angles and Euler angular velocities to Euler angular velocities as a function of $\dot{R}$ and Euler angles results in the elimination of Euler angular velocity terms. Next, using R, which is a function of Euler angles, the same method can be used to define Euler angles as functions of R. These Euler angles as function of R can then be implemented in the definitions for Euler angular velocities as function of $\dot{R}$ and Euler angles, to make Euler angular velocities as a function of R and $\dot{R}$. This completes the description of the used method. The following subsections specify the method for both determining the current attitude state and the required reference attitude state.

A.2.1 Current attitude

As mentioned, recall (2.1). For clarity, the resulting matrix is presented below

$$R = \begin{bmatrix} c\theta c\psi & s\phi s\theta c\psi - c\phi s\psi & c\phi s\theta c\psi + s\phi s\psi \\ c\theta s\psi & s\phi s\theta s\psi + c\phi c\psi & c\phi s\theta s\psi - s\theta c\psi \\ -s\theta & s\phi c\theta & c\phi c\theta \end{bmatrix}, \tag{A.6}$$

where it can be noticed that $R_{31} = -\sin\theta$. Rewriting this expression results in

$$\sin\theta = -R_{31}, \tag{A.7}$$

which also leads to the complementary expression for $\cos\theta$ using the Pythagorean identity. After implementing these expressions in (A.6), the following can be obtained

$$\cos\psi = \frac{R_{11}}{\sqrt{1 - R_{31}^2}}, \tag{A.8}$$

which again can be rewritten to an expression for $\sin\psi$ using the Pythagorean identity. Following the same method, the expressions for ψ are implemented in (A.6), together with the expressions for θ to obtain the following expressions for ϕ

$$\sin\phi = \frac{R_{32}}{\sqrt{1 - R_{31}^2}} \tag{A.9a}$$

$$\cos\phi = \sqrt{\frac{-1 + R_{31}^2 + R_{32}^2}{-1 + R_{31}^2}} = \sqrt{\frac{R_{33}^2}{1 - R_{31}^2}}. \tag{A.9b}$$

These expressions can be used to convert Euler angles to terms of $SO(3)$. For more information on how to retrieve the exact Euler angles, without the cos, sin and tan terms, refer to the next section

for a pseudo code.

From (A.6), find that

$$\dot{R} = \frac{\mathrm{d}}{\mathrm{dt}}R$$
$$= \begin{bmatrix} \dot{\psi}c\theta s\psi - \dot{\theta}c\psi s\theta & -c\phi c\psi(\dot{\psi} - \dot{\phi}s\theta) + s\phi(\dot{\theta}c\psi c\theta + s\psi(\dot{\phi} - \dot{\psi}s\theta)) & c\psi s\phi(\dot{\psi} - \dot{\phi}s\theta) + c\phi(\dot{\theta}c\psi c\theta + s\psi(\dot{\phi} - \dot{\psi}s\theta)) \\ \dot{\psi}c\psi c\theta - \dot{\theta}s\psi s\theta & -c\psi s\phi(\dot{\phi} - \dot{\psi}s\theta) + s\psi(\dot{\theta}c\theta s\phi + c\phi(-\dot{\psi} + \dot{\phi}s\theta)) & s\phi s\psi(\dot{\psi} - \dot{\phi}s\theta) + c\phi(\dot{\theta}c\theta s\psi + c\psi(-\dot{\phi} + \dot{\psi}s\theta)) \\ -\dot{\theta}c\theta & \dot{\phi}c\phi c\theta - \dot{\theta}s\phi s\theta & -\dot{\phi}c\theta s\phi - \dot{\theta}c\phi s\theta \end{bmatrix}. \tag{A.10}$$

From $\dot{R}_{31}$, note that

$$\dot{\theta} = -\frac{\dot{R}_{31}}{\cos\theta}, \tag{A.11}$$

with which, after substituting the new definition for $\dot{\theta}$, also $\dot{\phi}$ can be found

$$\dot{\phi} = \frac{\dot{R}_{32} - \dot{R}_{31}\sin\phi\tan\theta}{\cos\phi\cos\theta}. \tag{A.12}$$

Repeating the previous step, now implementing the new $\dot{\phi}$ definition leads to the following definition for $\dot{\psi}$

$$\dot{\psi} = \frac{\dot{R}_{21} - \dot{R}_{31}\sin\psi\tan\theta}{\cos\psi\cos\theta}. \tag{A.13}$$

Now the Euler angular velocities have been defined in terms of Euler angles and $\dot{R}$. To complete the conversion to $SO(3)$, implement the definitions from (A.7), (A.8) and (A.9) in the equations from (A.11), (A.12) and (A.13). To reconstruct the current attitude in $SO(3)$ from Euler angles, the opposite relations can be used, together with (A.6).

The conversions in (A.7), (A.8), (A.9), (A.11), (A.12) and (A.13) have been used to convert the decoupling terms needed for (4.4) from Euler angles to $SO(3)$ terms.

A.2.2 Reference attitude

To construct the necessary attitude references from $SO(3)$ to Euler angles for the tracking dynamics in Section 4.2.1, an opposite method is used. Now, instead of (A.6), the matrix from (4.10) is used to define the Euler angles resulting from the reference attitude matrix (4.8). Note that no yaw definition has to be derived here, since the reference attitude should be described using online pitch and roll angles.

First, note that the same relation between $\sin\theta$ and R_d as in (A.7) can be seen

$$\sin\theta = R_{d,31}, \tag{A.14}$$

where again $\cos\theta$ can be retrieved using the Pythagorean identity. The relations for ϕ this time are relatively straightforward as they can be retrieved directly, without implementing the expressions for θ:

$$\sin\phi = -R_{d,23} \tag{A.15a}$$
$$\cos\phi = R_{d,22}. \tag{A.15b}$$

These expressions can be used to convert $SO(3)$ terms to the Euler parametrization. For more information on how to retrieve the exact Euler angles, without the cos, sin and tan terms, refer to the next section for a pseudo code.

A.3 Pseudo codes

In this thesis, two similar, yet different, pseudo codes are used to define the tracking dynamics of (4.6c). First the pseudo code to define y_i is shown and subsequently the pseudo code for $y_{i,r}$.

Algorithm 1: Pseudo code to convert current attitude R to Euler angles.

Data: R
Result: rewrite the attitude R to Euler angles
[U,∼,V] = svd(R);
R = U*V' ; % To restore properties R: $\det(R) = 1$ and $R^\top R = I$.
if $||f_{d1}|| \neq 1$ **then**
  $\theta_1 = -\arcsin(R_{31})$;
  $\theta_2 = \pi - \theta_1$;
  $\phi_1 = \text{atan2}(R_{32}/\cos(\theta_1), R_{33}/\cos(\theta_1))$;
  if $\cos\phi_1 \geq 0$ **then**
    $\phi = \phi_1$;
    $\theta = \theta_1$;
    $\psi = \text{atan2}(R_{21}/\cos(\theta_1), R_{11}/\cos(\theta_1))$;
  else
    $\phi = \text{atan2}(R_{32}/\cos(\theta_2), R_{33}/\cos(\theta_2))$;
    $\theta = \theta_2$;
    $\psi = \text{atan2}(R_{21}/\cos(\theta_2), R_{11}/\cos(\theta_2))$;
  end
else
  Error: Gimbal lock.
end

Algorithm 2: Pseudo code to convert desired attitude R_d to Euler angles.

Data: f_d
Result: rewrite the desired attitude R_d to Euler angles
if $||f_{d,1}|| \neq 1$ **then**
  $\phi_1 = -\arcsin(f_{d2})$;
  $\phi_2 = \pi - \phi_1$;
  if $\cos(\phi_1) \geq 0$ **then**
    $\phi_d = \phi_1$;
  else
    $\phi_d = \phi_2$;
  end
  $\theta_d = \text{atan2}(f_{d1}/\cos\phi_d, f_{d3}/\cos\phi_d)$;
else
  Error: reference is singular.
end

Appendix B

Flat output trajectories

B.1 Derivation of rotation matrix for a given third column

Lefeber, van den Eijnden, and Nijmeijer (2017) derived the orientation of the drone based on the direction of their desired thrust vector. The orientation that followed ensured almost always correct tracking, but was arbitrary, since only its global position was tracked. The following method describes how to specify the heading direction as well. Meaning that an extra tracking coordinate has to be added in the reference tracking dynamics next to the global position $\rho_{\mathcal{I}}$, namely yaw rotations, denoted by ψ.

Hereto, the first step is to find a new expression for the (reference) rotation matrix. The rotation matrix from [1] consisted only of accelerations in x, y and z. As mentioned earlier, it is desired to include ψ. Therefore, it can firstly be noted that the third column of the rotation matrix can be rewritten to

$$\begin{aligned}\begin{bmatrix} r_1 \\ r_2 \\ r_3 \end{bmatrix} &= \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \\ &= \begin{bmatrix} \cos\psi\sin\theta\cos\phi + \sin\psi\sin\phi \\ \sin\psi\sin\theta\cos\phi - \cos\psi\sin\phi \\ \cos\theta\cos\phi \end{bmatrix}.\end{aligned}$$

From here, three separate expressions can be obtained. Namely those of r_1, r_2 and r_3. Now, using $\cos\phi \neq 0$, these expressions can be rewritten to:

$$\sin\theta = \frac{r_1\cos\psi + r_2\sin\psi}{\cos\phi} \tag{B.1a}$$

$$\sin\phi = r_1\sin\psi - r_2\cos\psi \tag{B.1b}$$

$$\cos\theta = \frac{r_3}{\cos\phi}. \tag{B.1c}$$

Substituting these expressions in (2.1) results in

$$
\begin{aligned}
R &= \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{r_3}{\cos\phi} & 0 & \frac{r_1\cos\psi + r_2\sin\psi}{\cos\phi} \\ 0 & 1 & 0 \\ -\frac{r_1\cos\psi - r_2\sin\psi}{\cos\phi} & 0 & \frac{r_3}{\cos\phi} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -r_1\sin\psi + r_2\cos\psi \\ 0 & r_1\sin\psi - r_2\cos\psi & \cos\phi \end{bmatrix} \\
&= \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{r_3}{\cos\phi} & \frac{(r_1\sin\psi - r_2\cos\psi)(r_1\cos\psi + r_2\sin\psi)}{\cos\phi} & r_1\cos\psi + r_2\sin\psi \\ 0 & \cos\phi & -r_1\sin\psi + r_2\cos\psi \\ -\frac{r_1\cos\psi + r_2\sin\psi}{\cos\phi} & \frac{r_3(r_1\sin\psi - r_2\cos\psi)}{\cos\phi} & r_3 \end{bmatrix} \\
&= \begin{bmatrix} \frac{r_3\cos\psi}{\cos\phi} & -\frac{(1-r_1^2)\sin\psi + r_1 r_2\cos\psi}{\cos\phi} & r_1 \\ \frac{r_3\sin\psi}{\cos\phi} & \frac{(1-r_2^2)\cos\psi + r_1 r_2\sin\psi}{\cos\phi} & r_2 \\ -\frac{r_1\cos\psi + r_2\sin\psi}{\cos\phi} & \frac{r_3(r_1\sin\psi - r_2\cos\psi)}{\cos\phi} & r_3 \end{bmatrix}.
\end{aligned} \tag{B.2}
$$

Using $\sin^2\phi + \cos^2\phi = 1$, $\cos\phi \geq 0$ and (B.1b) results in:

$$
\cos\phi = \sqrt{1 - (r_1\sin\psi - r_2\cos\psi)^2}. \tag{B.3}
$$

This in turn can be used to rewrite (B.2):

$$
R = \begin{bmatrix} \frac{r_3\cos\psi}{\sqrt{1-(r_1\sin\psi - r_2\cos\psi)^2}} & -\frac{(1-r_1^2)\sin\psi + r_1 r_2\cos\psi}{\sqrt{1-(r_1\sin\psi - r_2\cos\psi)^2}} & r_1 \\ \frac{r_3\sin\psi}{\sqrt{1-(r_1\sin\psi - r_2\cos\psi)^2}} & \frac{(1-r_2^2)\cos\psi + r_1 r_2\sin\psi}{\sqrt{1-(r_1\sin\psi - r_2\cos\psi)^2}} & r_2 \\ -\frac{r_1\cos\psi + r_2\sin\psi}{\sqrt{1-(r_1\sin\psi - r_2\cos\psi)^2}} & \frac{r_3(r_1\sin\psi - r_2\cos\psi)}{\sqrt{1-(r_1\sin\psi - r_2\cos\psi)^2}} & r_3 \end{bmatrix}. \tag{B.4}
$$

In only two cases it is possible that $\cos\phi = 0$ occurs: $r = \begin{bmatrix} -\sin\psi & \cos\psi & 0 \end{bmatrix}^\top$ and $r = \begin{bmatrix} \sin\psi & -\cos\psi & 0 \end{bmatrix}^\top$, where the former corresponds with $\phi = -\frac{\pi}{2}$ and the latter with $\phi = \frac{\pi}{2}$. In case of the latter, it thus occurs that

$$
R = \begin{bmatrix} \cos\theta\cos\psi & \sin\theta\cos\psi & \sin\psi \\ \cos\theta\sin\psi & \sin\theta\sin\psi & -\cos\psi \\ -\sin\theta & \cos\theta & 0 \end{bmatrix} = \begin{bmatrix} -r_2\cos\theta & -r_2\sin\theta & r_1 \\ r_1\cos\theta & r_1\sin\theta & r_2 \\ -\sin\theta & \cos\theta & 0 \end{bmatrix}, \tag{B.5}
$$

which implies that the pitch, θ, cannot be determined from the final column of R for a roll, ϕ, of $\frac{\pi}{2}$. From (B.1a) and (B.1c), one can see the problems occurring with $\phi = \frac{\pi}{2}$. Since it is necessary to know the orientation of the drone at any given time, a roll of $\frac{\pi}{2}$ is to be avoided.

B.2 Derivation of angular velocities

The next step in rewriting the reference coordinates is to express the angular velocities, ω, in terms of the newly defined rotation matrix R from (B.4).

Note that from (2.4c), it is known that $\dot{R}_{\mathcal{BI}} = R_{\mathcal{BI}} S(\omega_{\mathcal{B}})$. Differentiating (B.2) and premultiplying with itself results in $S(\omega_{\mathcal{B}})$. Using the definition in (2.5), the following can be obtained

$$
\omega_{\mathcal{B}} = \begin{bmatrix} \dot{\phi} - \sin\theta\dot{\psi} \\ \cos\theta\sin\phi\dot{\psi} + \cos\phi\dot{\theta} \\ \cos\theta\cos\phi\dot{\psi} - \sin\phi\dot{\theta} \end{bmatrix}. \tag{B.6}
$$

The terms $\dot{\theta}$ and $\dot{\phi}$, are undesired and therefore need to be rewritten in terms of other variables. By starting with differentiating (B.1c), a new definition for $\dot{\phi}$ can be derived

$$
\begin{aligned}
\dot{\phi} &= \frac{\frac{\partial}{\partial r_1}\sin\phi\dot{r}_1 + \frac{\partial}{\partial r_2}\sin\phi\dot{r}_2 + \frac{\partial}{\partial r_3}\sin\phi\dot{r}_3 + \frac{\partial}{\partial\psi}\sin\phi\dot{\psi} + \frac{\partial}{\partial\phi}\sin\phi\dot{\phi}}{\cos\phi} \\
&= \frac{\sin\psi}{\cos^2\phi}\dot{r}_1 - \frac{\cos\psi}{\cos^2\phi}\dot{r}_2 + \frac{r_1\cos\psi + r_2\sin\psi}{\cos^2\phi}\dot{\psi}.
\end{aligned} \tag{B.7}
$$

The $\dot{\theta}$-term can be derived from either $\cos\theta$ or $\sin\theta$, the following derivation is done through $\sin\theta$.

$$\dot{\theta} = \frac{\frac{\partial}{\partial r_1}\sin\theta\dot{r}_1 + \frac{\partial}{\partial r_2}\sin\theta\dot{r}_2 + \frac{\partial}{\partial r_3}\sin\theta\dot{r}_3 + \frac{\partial}{\partial\psi}\sin\theta\dot{\psi} + \frac{\partial}{\partial\phi}\sin\theta\dot{\phi}}{\cos\theta}.$$

Note that the new expression of $\dot{\phi}$ can be implemented here. This, combined with the definitions of $\sin\theta$ and $\cos\theta$, results in

$$\dot{\theta} = \frac{(-1+r_2^2)\cos\psi - r_1 r_2\sin\psi}{-r_3\cos^2\phi}\dot{r}_1 + \frac{r_1^2\sin\psi - \sin\psi - r_1 r_2\cos\psi}{-r_3\cos^2\phi}\dot{r}_2 + \frac{r_2(-1+r_1^2+r_2^2)\cos\psi - r_1(-1+r_1^2+r_2^2)\sin\psi}{-r_3\cos^2\phi}\dot{\psi}.$$

Knowing that each column from R is a unit vector, since it is a rotation matrix, it can be said that

$$r_1^2 + r_2^2 + r_3^2 = 1. \tag{B.8}$$

This results in

$$\begin{aligned}
\dot{\theta} &= -\frac{r_1^2\cos\psi + r_3^2\cos\psi + r_1 r_2\sin\psi}{-r_3\cos^2\phi}\dot{r}_1 - \frac{(r_2^2+r_3^2)\sin\psi + r_1 r_2\cos\psi}{-r_3\cos^2\phi}\dot{r}_2 - \frac{r_2 r_3^2\cos\psi - r_1 r_3^2\sin\psi}{-r_3\cos^2\phi}\dot{\psi} \\
&= \frac{r_3\cos\psi}{\cos^2\phi}\dot{r}_1 + \frac{r_1^2\cos\psi + r_1 r_2\sin\psi}{r_3\cos^2\phi}\dot{r}_1 + \frac{r_3\sin\psi}{\cos^2\phi}\dot{r}_2 + \frac{r_2^2\sin\psi + r_1 r_2\cos\psi}{r_3\cos^2\phi}\dot{r}_2 - \frac{r_3(r_1\sin\psi - r_2\cos\psi)}{\cos^2\phi}\dot{\psi} \\
&= \frac{r_3\cos\psi}{\cos^2\phi}\dot{r}_1 + \frac{r_3\sin\psi}{\cos^2\phi}\dot{r}_2 + \frac{(r_1\cos\psi + r_2\sin\psi(\dot{r}_1 r_1 + \dot{r}_2 r_2))}{r_3\cos^2\phi} - \frac{r_3(r_1\sin\psi - r_2\cos\psi)}{\cos^2\phi}\dot{\psi}.
\end{aligned}$$

Differentiation of (B.8) yields

$$\dot{r}_1 r_1 + \dot{r}_2 r_2 + \dot{r}_3 r_3 = 0, \tag{B.9}$$

which can be used to simplify $\dot{\theta}$ further to get

$$\begin{aligned}
\dot{\theta} &= \frac{r_3\cos\psi}{\cos^2\phi}\dot{r}_1 + \frac{r_3\sin\psi}{\cos^2\phi}\dot{r}_2 - \frac{(r_1\cos\psi + r_2\sin\psi)(\dot{r}_3 r_3))}{r_3\cos^2\phi} - \frac{r_3(r_1\sin\psi - r_2\cos\psi)}{\cos^2\phi}\dot{\psi} \\
&= \frac{r_3\cos\psi}{\cos^2\phi}\dot{r}_1 + \frac{r_3\sin\psi}{\cos^2\phi}\dot{r}_2 - \frac{r_1\cos\psi + r_2\sin\psi}{\cos^2\phi}\dot{r}_3 - \frac{r_3(r_1\sin\psi - r_2\cos\psi)}{\cos^2\phi}\dot{\psi}.
\end{aligned} \tag{B.10}$$

Combining equations B.7 and B.10 to rewrite (B.6), results in

$$\omega_{\mathcal{B}} = \begin{bmatrix} \frac{\dot{r}_1\sin\psi - \dot{r}_2\cos\psi}{\sqrt{1-(r_1\sin\psi - r_2\cos\psi)^2)}} \\ \frac{r_3\cos\psi}{\sqrt{1-(r_1\sin\psi - r_2\cos\psi)^2)}}\dot{r}_1 + \frac{r_3\sin\psi}{\sqrt{1-(r_1\sin\psi - r_2\cos\psi)^2)}}\dot{r}_2 - \frac{r_1\cos\psi + r_2\sin\psi}{\sqrt{1-(r_1\sin\psi - r_2\cos\psi)^2)}}\dot{r}_3 \\ -\frac{r_3\cos\psi(r_1\sin\psi - r_2\cos\psi)}{1-(r_1\sin\psi - r_2\cos\psi)^2)}\dot{r}_1 - \frac{r_3\sin\psi(r_1\sin\psi - r_2\cos\psi)}{1-(r_1\sin\psi - r_2\cos\psi)^2)}\dot{r}_2 + \frac{(r_1\sin\psi - r_2\cos\psi)(r_1\cos\psi + r_2\sin\psi)}{1-(r_1\sin\psi - r_2\cos\psi)^2)}\dot{r}_3 + \frac{r_3}{1-(r_1\sin\psi - r_2\cos\psi)^2)}\dot{\psi} \end{bmatrix}, \tag{B.11}$$

which can be written more compactly as

$$\omega_{\mathcal{B}} = \begin{bmatrix} \omega_{\mathcal{B},1} \\ \omega_{\mathcal{B},2} \\ \omega_{\mathcal{B},3} \end{bmatrix} = \begin{bmatrix} \frac{1}{r_3}(R_{2,1}\dot{r}_1 - R_{1,1}\dot{r}_2) \\ R_{1,1}\dot{r}_1 + R_{2,1}\dot{r}_2 + R_{3,1}\dot{r}_3 \\ -\frac{R_{3,2}}{r_3}\omega_{\mathcal{B},2} + \frac{r_3}{\cos^2\phi}\dot{\psi} \end{bmatrix}, \tag{B.12}$$

where $R_{i,j}$ denotes the element in row i and column j of the rotation matrix R as derived in (B.4).

B.3 Rewriting the reference dynamics

For the last step in rewriting the reference dynamics into the desired coordinates, $\rho_{\mathcal{I}} = \begin{bmatrix} x & y & z \end{bmatrix}^\top$ and ψ need to be a flat output (i.e., so that the state and input from these signals and their derivatives can be determined [16]). Therefore, some assumptions need to be made first. These assumptions are that

1. ρ and ψ are respectively four and two times differentiable at least,
2. $\ddot{x}^2 + \ddot{y}^2 + (g - \ddot{z})^2 \neq 0$ to assure positive thrust, and

3. if $\ddot{z} = g$, then $\ddot{x}\sin\psi - \ddot{y}\cos\psi \neq -\sqrt{\ddot{x}^2 + \ddot{y}^2}$ to avoid a roll, ϕ, of $\frac{\pi}{2}$.

Then, from the dynamics (2.4), one can write

$$\begin{aligned}\ddot{\rho}_{\mathcal{I}} &= \dot{R}_{\mathcal{BI}}\nu_{\mathcal{B}} + R_{\mathcal{BI}}\dot{\nu}_{\mathcal{B}} \\ &= ge_{3\mathcal{I}} - (f/m)R_{\mathcal{BI}}e_{3\mathcal{B}},\end{aligned} \tag{B.13}$$

which, using $f > 0$, results in

$$\begin{aligned}f &= \|fR_{\mathcal{BI}}e_{3\mathcal{B}}\| \\ &= m\left\|ge_{3\mathcal{I}} - \ddot{\rho}_{\mathcal{I}}\right\|_2 \\ &= m\sqrt{\ddot{x}^2 + \ddot{y}^2 + (g - \ddot{z})^2},\end{aligned} \tag{B.14}$$

where the properties $\|R\| = 1$ and $\|e_3\| = 1$ are used. Also note that $R_{\mathcal{BI}}e_{3_{\mathcal{B}}}$ can be defined from (B.13) as follows

$$R_{\mathcal{BI}}e_{3\mathcal{B}} = \frac{m}{f}(ge_{3\mathcal{I}} - \ddot{\rho}_{\mathcal{I}}), \tag{B.15}$$

where the definition of f can be substituted for (B.14), resulting in

$$\begin{aligned}R_{\mathcal{BI}}e_{3\mathcal{B}} &= \frac{ge_{3\mathcal{I}} - \ddot{\rho}_{\mathcal{I}}}{\|ge_{3\mathcal{I}} - \ddot{\rho}_{\mathcal{I}}\|} \\ &= \begin{bmatrix} \frac{-\ddot{x}}{\sqrt{\ddot{x}^2+\ddot{y}^2+(g-\ddot{z})^2}} \\ \frac{-\ddot{y}}{\sqrt{\ddot{x}^2+\ddot{y}^2+(g-\ddot{z})^2}} \\ \frac{g-\ddot{z}}{\sqrt{\ddot{x}^2+\ddot{y}^2+(g-\ddot{z})^2}} \end{bmatrix}.\end{aligned} \tag{B.16}$$

Now $R_{\mathcal{BI}}$ can be completed using (B.4). Also, using (B.12) $\omega_{\mathcal{B}}$ follows. Finally, $\tau_{\mathcal{B}}$ and $\nu_{\mathcal{B}}$ follow from (2.4a) and (2.4d). Note that the dynamics are now expressed in arbitrary $\rho_{\mathcal{I}}$ and ψ, as long as they satisfy the three constraints stated earlier, which is generally the case, since trajectories that do not meet the constraints are mostly not useful.